BAB 1 PENDAHULUAN

1.1 Pendahuluan

Seiring dengan berkembangnya teknologi orang semakin dimudahkan untuk menyelesaikan pekerjaannya, misalnya dalam hal pengolahan data. Pada pengolahan data terdapat isi *file* berupa teks misalnya surat formal. Biasanya dalam surat formal terdapat isi surat yang hanya boleh diketahui olek pihak tertentu dan bersifat sangat rahasia. Tetapi dengan semakin berkembangnya teknologi pada saat ini banyak pula ancaman yang terjadi pada *file* tersebut. Oleh sebab itu, dibutuhkan tingkat keamanan untuk melindungi *file* yang bersifat rahasia tersebut. Terdapat banyak cara untuk memberi keamanan terhadap data dari ancaman pihak yang tidak bertanggung jawab. Terdapat solusi untuk masalah ini yaitu dengan menggunakan teknik penyandian. Teknik penyandian ini dapat digunakan untuk mengamankan data yang diinginkan. Dengan adanya teknik ini data yang sangat rahasia dapat diamankan sedemikian rupa sehingga tidak dapat dibaca. Proses penyandian ini biasanya disebut dengan kriptografi.

Kriptografi merupakan ilmu yang mempelajari tentang algoritme yang digunakan untuk menyandikan data sehingga susah dibaca. Dalam teknik penyandian terdapat proses supaya data dapat diamankan yaitu proses enkripsi. Enkripsi merupakan proses untuk mengubah *plaintext* (data sebenarnya) menjadi *ciphertext* (penyandian). Enkripsi diperlukan supaya data yang rahasia menjadi dalam bentuk tulisan yang tidak dapat di baca, sehingga menyulitkan pihak luar yang ingin menyalahgunakan data tersebut. Dalam kriptografi juga terdapat proses dekripsi. Dekripsi merupakan proses mengubah *ciphertext* (penyandian) menjadi *plaintext* (data sebenarnya). Dekripsi diperlukan supaya pihak yang bersangkutan dapat membaca kembali data yang sudah dienkripsi. Dalam proses enkripsi maupun dekripsi membutuhkan sebuah *key* (kunci). Terdapat dua macam algoritme kriptografi yaitu kriptografi asimetris dan simetris. Dalam kriptografi asimetris memakai kunci yang berbeda untuk proses enkrìpsi dan dekripsi. Sedangkan kriptografi simetris memakai kunci yang sama untuk proses enkrìpsi dan dekripsi (Ariyus, 2008).

Pada penelitian sebelumnya dilakukan oleh Rachmawanto, dkk (2015) menggunakan tipe algoritme simetris yang artinya kunci yang digunakan dalam proses enkripsi dan dekripsi menggunakan kunci yang sama. Penelitian tersebut menggunakan algoritme Shift c*ipher* dan mendapatkan hasil bahwa algoritme ini mempunyai kehandalan dalam mengamankan data yang dilakukan pada sejumlah *file* dokumen. Hasil ekstraksi *file* telah berhasil dilakukan tanpa merusak *file* induk dan *file* pesan tanpa mengubah isi dan ukuran *file*. Tetapi algoritme Shift *cipher* merupakan algoritme kriptografi klasik yang memiliki kelemahan dalam pengamanan sebuah *file*, sehingga dibutuhkan algoritme baru untuk mengamankan sebuah *file.*

Berdasarkan permasalahan tersebut, peneliti mencoba memberikan penyelesaian untuk menjaga kerahasiaan sebuah *file* dari orang yang tidak berhak megetahuinya. Dalam merahasiakan sebuah *file* dibutuhkan juga *availability* (ketersediaan) data yang menggunakan konsep *secret sharing*. *Secret*

sharing digunakan untuk menjaga *file* agar tidak mudah hilang sehingga terdapat sejumlah *file* cadangan. *Secret sharing* memiliki dua proses yaitu proses *share* dan *reconstruct*. Proses *share* merupakan proses dimana data asli akan dipecah menjadi beberapa bagian, sedangkan proses *reconstruct* merupakan pengembalian data yang sudah dipecah menjadi satu. Maka dari itu, dibutuhkan algoritme baru yang dapat mengamankan *file* tersebut. Dibutuhkan juga algoritme yang dapat memecah *file* menjadi beberapa bagian, sehingga keamanan *file* dapat lebih terjaga.

Penelitian ini menggunakan algoritme SPECK *block cipher* untuk proses enkripsi dan dekripsinya. Algoritme SPECK diajukan publik pada Juni 2013 oleh sekelompok peneliti di Direktorat Penelitian US *National Security Agency's* (Beaulieu, Ray dkk, 2015). *Block cipher* digunakan untuk mengenkripsi dan mendekripsi informasi dalam blok ukuran tetap (Canavan, 2001). Dibutuhkan juga skema *secret sharing* yang digunakan untuk membagi data rahasia kepada pihak yang bersangkutan. Algoritme Shamir's *secret sharing* digunakan untuk memecah *file* menjadi beberapa bagian. Apabila ingin menyatukan kembali *file* yang sudah dipecah maka akan dilakukan proses *reconstruct* (Shamir, 1979).

Mengkaji dari penelitian yang ada, penulis menggunakan algoritme SPECK 128/128 dan Shamir's *secret sharing* pada *file* teks. Kombinasi dari kedua algoritme tersebut diharapkan dapat meningkatkan kerahasiaan dan ketersediaan *file* tersebut. Oleh karena itu, penelitian ini mengangkat tema "Implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks".

1.2 Rumusan masalah

Berdasarkan latar belakang yang sudah dikemukakan, maka rumusan masalah yang akan dibahas dalam penelitian ini adalah :

1. Bagaimana implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks?
2. Berapa waktu yang dibutuhkan untuk proses enkripsi, dekripsi serta *reconstruct* dengan algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks?
3. Berapa banyak pemakaian CPU dan RAM untuk algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks?

1.3 Tujuan

Berdasarkan masalah yang sudah dikemukakan, maka terdapat tujuan dari masalah tersebut yaitu :

1. Mengimplementasikan algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.
2. Mengetahui hasil waktu yang dibutuhkan untuk proses enkripsi, dekripsi serta *reconstruct* dengan algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.
3. Mengetahui banyaknya pemakaian CPU dan RAM untuk algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

1.4 Manfaat

1.4.1 Bagi Peneliti

Terdapat manfaat penelitian bagi peneliti itu sendiri yaitu :

1. Peneliti dapat mengetahui algoritme SPECK *block cipher* dan Shamir's *secret sharing* dalam proses enkripsi dan dekripsi pada *file* teks.
2. Peneliti dapat mengetahui hasil waktu yang dibutuhkan untuk proses enkripsi, dekripsi serta *reconstruct* dengan algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.
3. Peneliti dapat mengetahui banyaknya pemakaian CPU dan RAM untuk algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

1.4.2 Bagi Pihak Lain

Terdapat manfaat penelitian bagi pihak lain yaitu :

1. Dengan adanya penelitian ini diharapkan pihak lain dapat mengetahui algoritme SPECK *block cipher* dan Shamir's *secret sharing*.
2. Dengan adanya penelitian ini diharapkan pihak lain dapat mengetahui cara kerja algoritme SPECK *block cipher* dan Shamir's *secret sharing*.
3. Dengan adanya penelitian ini diharapkan dapat memberikan masukan terhadap pihak yang membutuhkan informasi tentang mengamankan *file* dengan algoritme SPECK *block cipher* dan Shamir's *secret sharing*.

1.5 Batasan masalah

Batasan dalam penelitian ini adalah sebagai berikut :

1. Algoritme membagi informasi atau kunci rahasia menggunakan Shamir's *secret sharing* dengan jumlah *share* maximal 5 dan rekrontruksi 4.
2. *File* dokumen yang digunakan dalam penelitian ini adalah *file* dokumen berformat PDF dengan menggunakan data berupa teks.
3. Input *file* masih menggunakan inputan manual dari program.
4. Sistem yang dibuat hanya untuk proses enkripsi dan *secret sharing* dalam satu laptop.

1.6 Sistematika pembahasan

Untuk mencapai tujuan yang telah dikemukakan di atas, maka sistematika penulisan tentang implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks adalah sebagai berikut:

BAB 1 Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian batasan penelitian, dan sistematika pembahasan tentang implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

BAB 2 Dasar Teori
Dasar teori menjelaskan tentang kajian pustaka dan dasar teori terkait dengan implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

BAB 3 Metodologi Penelitian
Metode Penelitian menjelaskan tentang metode yang digunakan untuk implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

BAB 4 Perancangan
Bab ini berisi tentang perancangan algoritme yang digunakan untuk implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

BAB 5 Implementasi .
Bab Implementasi menjelaskan tentang implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

BAB 6 Pengujian dan Analisis
Bab ini berisi tentang pengujian yang dilakukan untuk implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks beserta analisis dan hasil yang sudah dilakukan pada tahap implementasi.

BAB 7 Penutup
Bab ini berisi tentang kesimpulan dan saran pada implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan merupakan pembahasan tentang teori, konsep, model, metode, atau sistem literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Landasan kepustakaan diambil dari landasan teori dari berbagai sumber pustaka yang terkait dengan penelitian yang sedang dilakukan.

2.1 Kajian Pustaka

Adanya penelitian yang didasarkan pada studi literatur pada beberapa penelitian sebelumnya yang pernah dilakukan yang berkaitan tentang Implementasi algoritme SPECK *block cipher* dan Shamir's *secret sharing* pada *file* teks. Sumber kajian pustaka dalam penelitian yang terdapat pada publikasi penelitian terdahulu seperti *paper*, jurnal, skripsi. Studi kepustakaan ini dilakukan dengan tujuan untuk memperkuat pemahaman domain permasalahan dan juga untuk mencari cara penyelesaian masalah terbaik.

Algoritme Simon dan SPECK diajukan publik pada Juni 2013 oleh sekelompok peneliti di Direktorat Penelitian US National Security Agency's. Desain pekerjaan ini dimulai tahun 2011, dan jumlah signifikan kriptoanalisis dilakukan (tidak hanya oleh para desainer, tetapi oleh banyak orang lain di perusahaan) dalam waktu yang mengarah ke penerbitan. Hasil ini membuktikan bahwa algoritme ini aman. Simon *block cipher* dengan n-bit kata (dan karenanya blok 2n-bit) dinotasikan Simon 2n, dimana n diharuskan untuk menjadi 16, 24, 32, 48, atau 64. Simon 2n dengan m-kata (mn bit) kunci akan disebut sebagai Simon2n / mn. Sebagai contoh, Simon64 / 128 mengacu pada versi Simon yang bekerja pada blok plaintext 64 bit dan menggunakan kunci 128-bit. Notasi untuk SPECK adalah analog. Penulis menggunakan notasi berikut untuk operasi pada kata-kata n-bit, di mana nilai n harus dipahami dari konteks : *bitwise* $\oplus$ (XOR), *bitwise* & (AND), mod 2n, *addition* (+), left circular shift, Sj, by j bits. SPECK 2n *round function* dipetakan dalam $(x, y) = ((S^{-\alpha}x + y) \oplus k, S^{\beta}\, y \oplus (S^{-\alpha}x + y) \oplus k$. Dimana x dan y adalah jumlah *n*-bit, dan k adalah kunci bulat. Parameter dan masing - masing 8 dan 3, kecuali pada kasus SPECK 32/ 64 (Beaulieu, Ray dkk, 2015).

Algoritme Shamir's s*ecret sharing scheme* dalam *paper* ini menunjukkan bagaimana membagi data D dalam potongan n dengan sedemikian rupa sehingga D mudah direkonstruksi dari setiap potongan k, tetapi bahkan pengetahuan lengkap tentang k - 1 buah sama sekali tidak ada informasi tentang D. Teknik ini memungkinkan konstruksi kunci yang kuat skema manajemen untuk sistem kriptografi dapat berfungsi dengan aman dan andal bahkan ketika kemalangan menghancurkan setengah bagian dan pelanggaran keamanan mengekspos semua kecuali satu dari sisa potongan (Shamir, 1979).

Tabel 2.1 Kajian Pustaka

No	Nama Peneliti, Tahun, dan Judul	Persamaan	Perbedaan	
			Penelitian sebelumnya	Penelitian penulis
1	Eko Hari Rachmawanto dan Christy Atika Sari, 2015, "Keamanan *File* Menggunakan Teknik Kriptografi Shift *Cipher*"	Menggunakan objek *file* teks	Implementasi pada file txt, doc, dan docx menggunakan algoritme Shift *cipher*.	Implementasi pada *file Portable Document Format* (PDF) menggunakan algoritme SPECK *block cipher*.
2	Adrian-Vasile Duka dan Bela Genge, 2017, "*Implementation of SIMON and* SPECK *lightweight Block Ciphers on Programmable Logic Controllers*""	Menggunakan algoritme SPECK *block cipher*	Implementasi pada *Programmable Logical Controller* (PLC) menggunakan algoritme Simon dan SPECK *block cipher*.	Implementasi pada *file Portable Document Format* (PDF) menggunakan algoritme SPECK *block cipher*.
3.	Mustafa Ulutas, Guzin Ulutas, Vasif V. Nabiyev, 2010, "*Medical image security and EPR hiding using* Shamir's *secret sharing scheme*"	Menggunakan konsep *secret sharing*.	Implementasi pada *Electro Patient Record* "EPR" dengan steganografi atau gambar medis menggunakan Shamir's *secret sharing*.	Implementasi pada *file Portable Document Format* (PDF) menggunakan algoritme Shamir's *secret sharing*.

2.2 Dasar Teori

2.2.1 Kriptografi

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar tetap aman. "*Crypto*" berarti "*secret*" (rahasia) dan (*graphy*) berarti "*writing*" (tulisan). Para pelaku atau praktisi kriptografi disebut *cryptographers*. Sebuah Algoritme kriptografi disebut *chipper* dan merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi (Ariyus, 2008). Ketika suatu pesan ditranster dan suatu tempat ke tempat lain. Isi dari pesan tersebut kemungkinan dapat disadap oleh pihak lain. Untuk menjaga keamanan

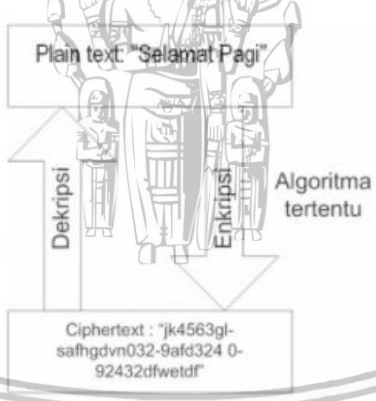
pesan, pesan tersebut dapat diacak (*screamble*) atau diubah menjadi kode yang tidak dapat dimengerti oleh orang lain (Drs. Ario Suryo Kusumo, 2004).

2.2.2 Tujuan Kriptografi

Terdapat beberapa tujuan dari kriptografi yaitu:

- *Confidentiality* kerahasiaan) bertujuan untuk melindungi identitas pemakai atau isi pesan supaya tidak dapat dibaca oleh orang lain yang tidak berhak.
- Data *Integrity* digunakan untuk melindungi pesan agar tidak diubah oleh orang lain.
- *Availability* bertujuan untuk menjamin ketersediaan sumber data.
- *Authentication* bertujuan untuk menjamin keaslian pesan.
- *Non repudiation* bertujuan untuk membuktikan suatu pesan berasal dari seseorang, apabila ia menyangkal mengirim pesan tersebut.

Dalam dunia kriptografi, pesan yang akan dirahasiakan disebut *plaintext*. Pesan yang sudah diacak disebut *ciphertext*. Proses untuk mengkonversi *plaintext* menjadi *ciphertext* disebut enkripsi. Proses untuk mengembalikan *plaintext* dan *ciphertext* disebut dekripsi. Algoritme kriptografi *ciphers* adalah fungsi-fungsi matematika yang digunakan untuk melakukan enkripsi dan dekripsi. Diperlukan kunci yaitu kode untuk melakukan enkripsi dan dekripsi (Drs. Ario Suryo Kusumo, 2004).



Gambar 2.1 Ilustrasi proses enkripsi dan dekripsi

Sumber: (Tim EMS, 2015)

2.2.3 Enkripsi

Enkripsi merupakan hal yang sangat penting dalam kriptografi. Enkripsi merupakan cara pengamanan data yang dikirimkan sehingga terjaga kerahasiaannya. Pesan pasti disebut *plaintext* (teks-biasa), yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi dapat diartikan dengan *cipher* atau kode. Sama halnya dengan tidak mengerti sebuah kata maka kita akan melihatnya di dalam kamus atau daftar istilah. Beda halnya dengan enkripsi, untuk mengubah teks-biasa ke bentuk teks-kode kita gunakan algoritme yang dapat mengkodekan data yang diinginkan (Ariyus, 2008).

2.2.4 Dekripsi

Dekripsi merupakan kebalikan dan enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya. Algoritme yang digunakan untuk dekripsi tentu berbeda dengan yang digunakan untuk enkripsi (Ariyus, 2008).

2.2.5 *Key*

Kunci adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian. yaitu kunci rahasia (*private key*) dan kunci umum (*public key*) (Ariyus, 2008).

2.2.6 Macam – Macam Algoritme Kriptografi

Algoritme kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya (Ariyus, 2008) :

- **Algoritme Simetri**

  Algoritme ini sering disebut dengan algoritme klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Algoritme ini sudah ada sejak lebih dari 4000 tahun yang lalu. Bila mengirim pesan dengan menggunakan Algoritme ini. Penerima pesan harus diberitahu kunci dari pesan tersebut agar dapat mendekripsikan pesan yang dikirim. Keamanan dan pesan yang menggunakan algoritme ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan. Algoritme yang memakai kunci simetri di antaranya adalah:

  1. *Data Encryption Standard* (DES),
  2. RC2, RC4, RC5,RC6,
  3. *International Data Encryption Algorithm* (IDEA),
  4. *Advanced Encryption Standard* (AES),
  5. *One Time Pad* (OTP),
  6. AS, dan lain sebagainya.

- **Algoritme Asimetri**

  Algoritme asimetri sering juga disebut dengan algoritme kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada Algoritme asimetri kunci terbagi menjadi dua bagian. yaitu:

  1) Kunci umum (*public key*) Kunci yang boleh semua orang tahu (dipublikasikan).
  2) Kunci rahasia (*private key*) Kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

  Kunci tersebut berhubungan satu sama lain. Dengan kunci publik orang dapat mengenkripsi pesan tetapi tidak dapat mendekripsinya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut. Algoritme yang memakai kunci publik di antaranya adalah:

  1. *Digital Signature Algorithm* (DSA),
  2. RSA,

3. *Diffie-Heliman* (DH),
4. *Elliptic Curve Cryptography* (ECC),
5. Kriptografi Quantum, dan lain sebagainya.

- ***Hash Function***

  Fungsi Hash sering disebut dengan fungsi Hash satu arah (*one-way function*), *message digest*, *fingerprint*, fungsi kompresi dan *message authentication code* (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Fungsi Hash biasanya diperlukan bila ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda bahwa pesan tersebut benar-benar berasal dan orang yang diinginkan.

2.2.7 SPECK *Block Cipher*

Algoritme SPECK adalah salah satu algoritme *block cipher* yang telah dirilis oleh National Security Agency (NSA) pada Juni 2013. Karena *block cipher*, maka algoritme SPECK menggunakan input (masukan) berupa blok (kumpulan bit) bukan bit. SPECK termasuk aplikasi yang ringan dan sangat baik dijalankan di dalam *software*. SPECK dirancang untuk dapat berjalan baik di *software* maupun *hardware*, terutama di dalam *microcontroller*. Algoritme ini memiliki besar kunci dan blok yang berbeda-beda tergantung pada kasus yang dihadapi. Satu blok akan terdiri 2 word. 1 word dapat berukuran 16, 24, 32, 48, atau 64 bit (Beaulieu, Ray dkk, 2015). Parameter SPECK bisa dilihat pada Tabel 2.2

Tabel 2.2 Parameter SPECK

Block Size (bits)	***Key Size (bits)***	***Rotation α***	***Rotation β***	***Rounds***
32	64	7	2	22
48	72	8	3	22
	96			23
64	96	8	3	26
	128			27
96	96	8	3	28
	144			29
128	128	8	3	32
	192			33
	256			34

2.2.8 Fungsi *Key Schedule*

Key schedule pada algoritme SPECK digunakan untuk membuat kunci setiap *round*. Kunci yang telah dibuat akan digunakan di fungsi *round* setiap *round*. Pada proses ini terdapat 2 variabel yang akan diproses, yaitu *k*i dan *li*.

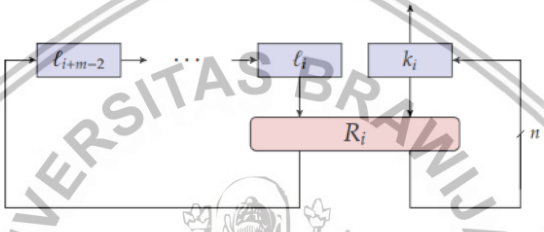
Untuk mendapatkan nilai *li* terdapat rumus seperti pada Persamaan 2.1.

$$l_{i-m-1} = (k_i + S^{-\alpha} l_i) \oplus \mathrm{i} \qquad (2.1)$$

Pada Persamaan 2.1, nilai m adalah nilai *word* pada *key* (tergantung jenis SPECK yang digunakan). Nilai i adalah *round* saat ini. Nilai i akan terus bertambah dari 0 sampai jumlah *round* sesuai jenis SPECK yang digunakan. Nilai l_{i-m-1} didapatkan dengan cara melakukan *addition modulo* 2_n antara k dengan l_i yang telah dilakukan rotasi ke kanan sejumlah α bit. Kemudian hasilnya akan dilakukan operasi XOR dengan nilai i. Untuk mendapatkan nilai k_i, dilakukan rumus seperti Persamaan 2.2.

$$k_{i+1} = S^{\beta} k_i \oplus l_{i-m-1} \tag{2.2}$$

Pada Persamaan 2.2, nilai k_{i+1} didapatkan dengan cara melakukan operasi XOR antara k_i yang telah dilakukan rotasi ke kiri sejumlah β bit dengan nilai l_{i-m-1} yang didapat dari Persamaan 2.1. Kedua rumus ini akan dilakukan perulangan jumlah *round* dari jenis SPECK yang digunakan. Ilustrasi gambaran proses *key schedule* pada algoritme SPECK dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi *Key Schedule* Pada Algoritme SPECK
Sumber : (Beaulieu, Ray dkk, 2015)

2.2.9 Fungsi *Round*

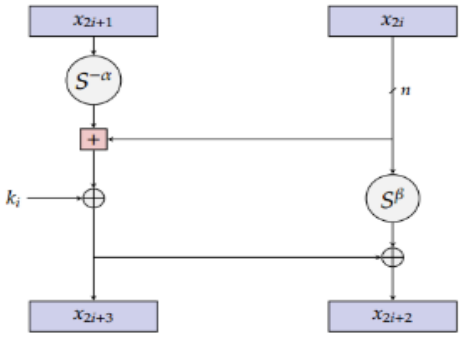
Operasi yang dilakukan dalam fungsi *round* ada 3, yaitu operasi *addition modulo* 2^n, rotasi ke kiri atau ke kanan sejumlah bit tertentu, dan XOR. Operasi *addition modulo* 2^n sama seperti melakukan operasi AND atau pertambahan per bit (Beaulieu, Ray dkk, 2015). Pada fungsi *round* ini terdapat rumus enkripsi dan dekripsi yang dipakai pada algoritme SPECK. Proses enkripsi terdapat pada Persamaan 2.3.

$$R_k(x,y) = ((S^{-\alpha}x + y) \oplus \mathrm{k}, S^{\beta}y \oplus \big((S^{-\alpha}x + y) \oplus \mathrm{k}\big)) \tag{2.3}$$

Pada persamaan 2.3, x adalah *plaintext* pertama dan y adalah *plaintext* kedua. X didapatkan dari melakukan *addition modulo* 2^n antara hasil x yang dilakukan rotasi ke kanan sejumlah α bit dan y. Kemudian hasil tersebut dilakukan operasi XOR terhadap *k* (key). Sedangkan Y didapatkan dari melakukan operasi rotasi ke kiri sejumlah ß bit pada y. Kemudian hasil tersebut dilakukan operasi XOR terhadap x yang telah dimasukan pada Persamaan 2.3. Proses dekripsi dilakukan dengan persamaan berikut

$${R_k}^{-1}(x,y) = (S^{\alpha}((x \oplus \mathrm{k}) - S^{\beta}(x \oplus y)), S^{-\beta}(x \oplus \mathrm{y})) \tag{2.4}$$

Pada rumus 2.4, x didapatkan dengan melakukan operasi XOR pada x dengan k (key). Hasil tersebut akan dilakukan rotasi ke kiri sejumalah α bit. Kemudian akan dikurangkan dengan hasil dari y yang sudah diproses pada Persamaan 2.4. Sedangkan y didapatkan dari melakukan operasi XOR antara x dan y. Kemudian hasilnya akan dirotasi ke kanan sejumlah β bit. Kedua proses ini akan diulang sejumlah *round* yang sesuai pada jenis SPECK yang digunakan. Ilustrasi gambaran fungsi *round* pada SPECK dapat dilihat pada Gambar 2.3.



Gambar 2.3 Ilustrasi fungsi *round* pada algoritme SPECK
Sumber : (Beaulieu, Ray dkk, 2015)

2.2.10 Shamir's *Secret Sharing*

Dalam metode *secret sharing* ada seorang pembagi dan n pemegang bagian (*shares*) yang disebut *participants*. Pembagi memberikan rahasia kepada pemegang bagian jika suatu syarat tertentu terpenuhi. Syarat yang dimaksud adalah struktur akses, yaitu aturan tentang siapa saja yang mendapat otorisasi untuk membentuk berita rahasia itu kembali. Dalam suatu kelompok berlaku nilai ambang (*threshold value*), yaitu jumlah minimal *participants* yang dibutuhkan dalam suatu kelompok untuk membangkitkan suatu berita rahasia. Metode ini tercipta disebabkan karena, untuk menjaga keamanan kunci hasil dari sistem kriptografi agar tidak hilang, disarankan untuk membuat sejumlah kunci cadangan (Lisa Chandra, 2012). Terdapat Interpolasi Lagrange diterapkan untuk mendapatkan fungsi polinomial $P(x)$ berderajat tertentu yang melewati sejumlah titik data. Misalnya, akan dicari fungsi polinomial berderajat satu yang melewati dua buah titik yaitu (x_0, y_0) dan (x_1, y_1).Interpolasi polinomial Lagrange dapat diturunkan dari persamaan Newton. Bentuk umum interpolasi polinomial Lagrange order *n* bisa dilihat pada Persamaan (2.5):

$$F_n(x) = \sum_{i=0}^{n} L_i\ (x)\ f(x_i) \tag{2.5}$$

Dengan

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq 1}}^{n} \frac{x-x_j}{x_i-x_j} \tag{2.6}$$

Penerapan Algoritme LaGrange Interpolating Polynomial pada Secret Sharing:

- Pilih suatu bilangan prima p yang harus lebih besar dari semua kemungkinan nilai secret (*plaintext*) M dan juga lebih besar dari jumlah n *participants*. p ini akan dijadikan modulus untuk semua perhitungan. p harus bilangan prima untuk memastikan tiap bilangan memiliki *invers*.
- Selanjutnya, pilih $t-1$ buah bilangan bulat dalam modulus p secara acak, misalkan $s_1, s_2, s_3 \ldots s_{t-1}$ dan bentuk suatu polinomial seperti pada persamaan 2.7 :

$$f(x) \equiv M + s_i x + s_2 x^2 + \cdots + s_{t-1} x^{t-1} (Mod\ P) \tag{2.7}$$

- Untuk n *participants*, tentukan n bilangan bulat berbeda dalam modulus p, misal $x_1, x_2, \ldots x_n\ (mod\ p)$ dan setiap orang memperoleh *share* (x_i, y_i) dimana $y_i \equiv f_i\ (x_i)\ (mod\ p)$

- Misalkan t orang *participants* akan merekonstruksi M, dengan share masing-masing (x_1, y_1) , $(x_2, y_2), \ldots . (x_t, y_t)$. Setelah itu substitusikan setiap (x_k, y_k) ke dalam polinomial $f(x)$ yang berarti jumlah *share* minimum 2 seperti Persamaan 2.8.

$$y_k \equiv M + s_1 x_k{}^1 + \cdots + s_{t-1} x_k{}^{t-1} \ (mod\ p)\ , 1 \leq k \leq t \qquad (2.8)$$

- Selesaikan system persamaan di atas untuk memperoleh $s_0 = M$ dengan menggunakan LaGrange. Dalam implementasi skema ini, nilai prima p tidak perlu rahasia, tetapi polinom $f(x)$ harus dirahasiakan.

2.2.11 Kolmogorov Smirnov

Kolgomorov Smirnov dilakukan untuk menguji normalitas. Uji normalitas adalah salah satu uji asumsi klasik yang bertujuan untuk membuktikan bahwa data yang akan diuji berdistribusi normal. Jika nilai pada baris Kolmogorov Smirnov Z bernilai di atas 0,05 maka distribusi data dinyatakan memenuhi asumsi normalitas, dan jika nilainya di bawah 0,05 maka diinterpretasikan sebagai tidak normal (Dicky Pratama & Herry Septriadi, 2016).

2.2.12 Anova

Untuk menganalisis data yang diperoleh menggunakan ANOVA (*Analysis Of Varian*), yaitu suatu metode atau salah satu uji hipotesis untuk melakukan pengujian terhadap interaksi antar dua faktor dalam suatu percobaan dengan membandingkan rata-rata dari lebih dua sampel. One way anova yaitu analisis ragam satu arah yang merupakan suatu prosedur untuk menguji rata- rata atau pengaruh perlakuan dari beberapa populasi (lebih dari dua) dari suatu percobaan yang menggunakan satu faktor, dimana satu faktor tersebut memiliki dua atau lebih level (Dino Caesaron, 2015). Untuk menguji apakah hipotesis dari penelitian akan menolak Ho atau H1,dengan ketentuan berikut :

a. Jika Fo > F tabel maka tolak Ho dan terima H1
b. Jika Fo < F tabel maka tolak H1 dan terima Ho

2.2.13 Kruskal-Wallis

Uji Kruskal-Wallis adalah uji yang sangat berguna untuk menentukan apakah k sampel independen berasal dari populasi-populasi yang berbeda. Hipotesis yang ada dalam uji Kruskal- Wallis adalah:

- H1 : Ke-k populasi memiliki median yang sama.
- H0 : Tidak semua dari ke-k populasi memiliki median yang sama.

Apabila keputusan yang diambil adalah menolak H1, maka kesimpulan yang diperoleh adalah tidak semua dari ke-k populasi memiliki median yang sama atau dengan kata lain tidak semua populasi asal sampel sama (Nawangsari, 2013).

2.2.14 Post Hoc Test

Uji Post Hoc adalah uji untuk membandingkan perbedaan mean antara 2 kelompok. Uji ini dilakukan untuk membandingkan antar kelompok (Ermawati, 2010). Penentuan uji :

- Apabila p > 0,05 berarti H0 diterima
- Apabila p < 0,05 berarti H0 ditolak

2.2.15 *Avalache Effect*

Avalanche Effect merupakan perubahan satu bit pada *plaintext* atau *key* yang menyebabkan perubahan yang signifikan terhadap *ciphertext*. Algoritme dikatakan baik jika nilai AE mempunyai perubahan satu bit saja pada input menghasilkan perubahan sekitar setengah jumlah bit pada output-nya. Maksimal nilai AE adalah 50%. Salah satu fungsi dari AE adalah untuk melihat tingkat keamanan suatu algoritme kriptografi (Endro Ariyanto, Trisya Indah Pravitasari, Setyorini, 2008). Rumus *Avalanche Effect* bisa dilihat pada Persamaan 2.9.

$$Avalanche\ Effect\ (AE) = \frac{\sum bit\ berubah}{\sum bit\ total}\ x100\% \quad (2.9)$$

BAB 3 METODOLOGI

Berikut adalah metodologi penelitian yang digunakan untuk Implementasi Algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing* Pada *File* Teks yaitu sebagai berikut :

3.1 Identifikasi Masalah

Identifikasi masalah merupakan pernyataan antara sebuah variabel dengan variabel lain yang memiliki hubungan dalam suatu fenomena. Dari latar belakang yang telah dibahas pada Bab 1, maka identifikasi masalah yang akan dijadikan penelitian adalah sebagai berikut :

1. *Confidentiality* (kerahasiaan)
2. *Secret Sharing*
3. *Availability* (ketersediaan)

3.2 Studi Kepustakaan

Mempelajari kepustakaan dari beberapa bidang ilmu yang berhubungan dengan Implementasi Algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing* Pada *File* Teks, diantaranya :

1. Ilmu Kriptografi
2. Algoritme SPECK *Block Cipher*
3. Algoritme Shamir's *Secret Sharing*

3.3 Spesifikasi Kebutuhan

Spesifikasi kebutuhan digunakan untuk memaparkan tentang kebutuhan yang berhubungan dengan penelitian Implementasi Algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing* Pada *File* Teks, diantaranya :

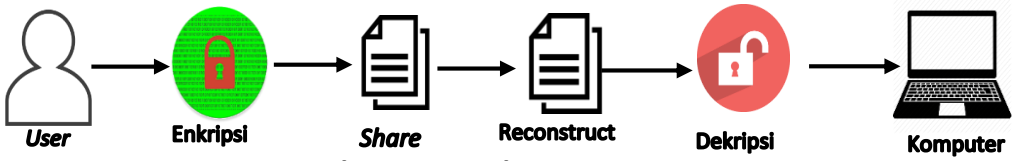
1. Aplikasi yang digunakan adalah NetBeans 8.0 dan JDK
2. Bahasa Pemrograman yang digunakan adalah Java
3. *File* yang digunakan adalah *file* PDF dan txt
4. *Library* yang digunakan adalah Itext-PDF 5.5.4

3.4 Metode Penelitian

Metode penelitian merupakan cara untuk mencari dan mendapatkan data. Metode penelitian yang digunakan dalam penelitian ini adalah metode implementatif (*developer*).

3.5 Perancangan

Perancangan digunakan untuk menggambarkan sistem yang akan dibuat. Dengan adanya perancangan maka, kebutuhan sistem akan lebih mudah untuk diidentifikasi secara jelas dan rinci. Pada tahap ini akan menjelaskan tentang kebutuhan sistem serta alur kerja pada setiap proses pada sistem. Gambaran umum sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 Gambaran Umum Sistem

Dalam melakukan enkripsi data, *user* memilih data yang akan dienkripsi. Selanjutnya data akan dienkripsi dengan algortitme SPECK *block cipher*. Setelah proses enkripsi selesai terdapat proses *share* yaitu memecah data hasil enkripsi menjadi beberapa bagian yang berbeda dengan menggunakan algoritme Shamir's *secret sharing*. Terdapat proses *reconstruct* yang bertujuan untuk mengembalikan data yang sudah dipecah menjadi satu dengan menggunakan algoritme Shamir's *secret sharing*. Dalam mengembalikan data semula, awalnya *user* memilih data *share* yang sudah tersimpan di komputer. Setelah itu data akan masuk dalam proses *reconstruct*. Terdapat minimal dua penggabungan (*reconstruct*) data yang sudah dipecah. Proses *reconstruct* menggunakan algoritme Shamir's *Secret Sharing,* sedangkan proses dekripsi menggunakan algoritme SPECK *Block Cipher*.

3.6 Implementasi

Implementasi merupakan pelaksanaan dari sebuah sistem yang sudah dirancang. Tujuan implementasi yaitu untuk mengkaji rangkaian sistem yang sudah dibuat. Terdapat tiga implementasi dalam penelitian ini, yaitu sebagai berikut :

1. Implementasi algoritme SPECK *Block Cipher*

   Pada tahap ini, algoritme SPECK *Block Cipher* diimplementasikan ke dalam bahasa java. Algoritme yang digunakan adalah SPECK 128/128 yang digunakan untuk enkripsi dan dekripsi *file*.
2. Implementasi algoritme Shamir's *Secret Sharing*

   Pada tahap ini, algoitma Shamir's *Secret Sharing* diimplementasikan ke dalam bahasa java. Algoritme ini digunakan untuk memecah *file* (*share*) menjadi beberapa bagian dan digunakan untuk menggabungkan *file* (*reconstruct*).
3. Implementasi penyimpanan dan pengambilan *file* ke dalam laptop

   Setelah *file* melewati proses enkripsi dan dekripsi, maka pada tahap ini *file* akan tersimpan ke dalam laptop. Setelah *file* tersimpan, maka *file* akan diambil lagi untuk proses *reconstruct*.

3.7 Pengujian dan Analisis

Pengujian digunakan untuk menguji sistem yang sudah dibuat serta untuk mengetahui kualitas dari sistem tersebut. Berikut adalah beberapa pengujian yang akan dilakukan :

1. Pengujian enkripsi dan dekripsi dengan algoritme SPECK *Block Cipher.*

Pengujian ini bertujuan untuk mengetahui *file* tersebut berhasil dienkripsi dan dekripsi dengan algoritme SPECK *Block Cipher*. Pengujian ini dinyatakan berhasil apabila *file* yang dienkripsi berubah menjadi data acak. Pengujian ini dinyatakan berhasil apabila *file* yang sudah menjadi data acak dapat berubah menjadi data asli.

2. Pengujian *test vector* dengan algoritme SPECK *Block Cipher.*
   Pengujian ini bertujuan untuk menyesuaikan nilai *test vector* dari *paper* asli SPECK dengan program yang dibuat pada implementasi SPECK 128/128. Pengujian ini dinyatakan berhasil apabila keluaran program sama dengan nilai *test vector* dari *paper* asli SPECK.
3. Pengujian *share* dan *reconstruct file* dengan algoritme Shamir's *Secret Sharing*.
   Pengujian ini bertujuan untuk mengetahui *file* yang akan melewati proses *share* dan *reconstruct* dapat berjalan dengan baik dan sesuai dengan program yang dibuat. Pengujian *share file* dinyatakan berhasil apabila *file* dapat terbagi menjadi beberapa bagian. Pengujian *reconstruct* dinyatakan berhasil apabila *file* yang tadinya terbagi dapat disatukan kembali menjadi *file* asli.
4. Pengujian waktu
   Pengujian ini bertujuan untuk mengetahui waktu proses enkripsi, dekripsi, *share*, *reconstruct*, proses enkripsi sampai *share*, proses *reconstruct* sampai dekripsi, *wirite file*, *read file* pada *file* teks. Pengujian ini akan dilakukan dengan menggunakan jumlah *share* dan *reconstruct file* teks yang berbeda.
5. Pengujian banyaknya pemakaian CPU dan RAM
   Pengujian ini bertujuan untuk mengetahui banyaknya pemakaian CPU dan RAM saat menjalankan sistem.

3.8 Kesimpulan dan Saran

Penarikan kesimpulan dilakukan setelah pembahasan telah selesai dilakukan. Penarikan kesimpulan digunakan untuk menjawab dari rumusan masalah yang sudah ditetapkan. Kesimpulan yang dapat diambil yaitu tentang Implementasi Algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing* Pada *File* Teks. Sedangkan saran dibutuhkan untuk memberi masukan dari penelitian yang telah dilakukan.

BAB 4 PERANCANGAN

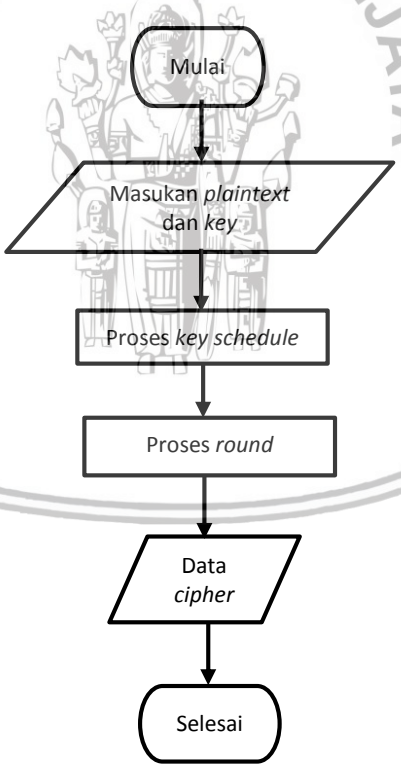
Bab empat menjelaskan tentang peranan perancangan sistem yang akan digunakan sebagai rujukan pada tahapan implementasi. Bab ini menjelaskan tentang perancangan sistem yang sudah di terapkan dalam *flowchart* mulai dari algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing*.

4.1 Perancangan Algoritme

Perancangan algoritme menjelaskan tentang penelitian dengan menggunakan algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing*.

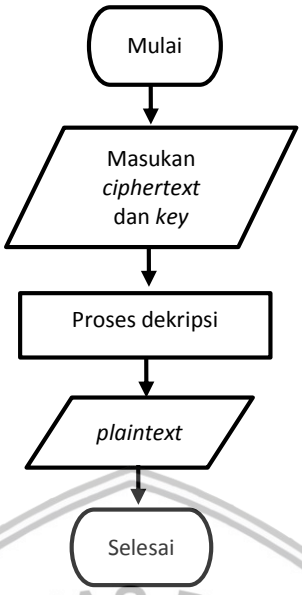
4.1.1 Algoritme SPECK *Block Cipher*

Berikut merupakan *flowchart* perancangan sistem tentang enkripsi Pada implementasi SPECK 128/128 yang digunakan dalam sistem ini. Pertama user memasukan *plaintext* dan *key*, setelah itu akan masuk pada proses *key schedule*. Kemudian masuk ke dalam proses *round*. Setelah melewati proses *key schedule* dan *round*, maka data awal tadi akan menjadi data *cipher*. Gambaran sistem dapat dilihat pada Gambar 4.1



Gambar 4.1 Perancangan Enkripsi Algoritme SPECK *Block Cipher*

Berikut merupakan *flowchart* perancangan sistem tentang dekripsi algoritme SPECK *Block Cipher* yang digunakan dalam sistem ini. Pertama user memasukan *ciphertext*, setelah itu akan masuk pada proses dekripsi dan akan menjadi *plaintext* atau data asli. Gambaran sistem dapat dilihat pada Gambar 4.2.



Gambar 4.2 Perancangan Dekripsi Algoritme SPECK *Block Cipher*

Dari alur sistem enkripsi dan dekripsi algoritme SPECK memiliki dua parameter yaitu *plaintext* dan *key schedule*. Dalam algoritme SPECK terdapat dua masukan yaitu *plaintext* dan *key*. Berikut adalah masukan *plaintext* dan *key* :

- *Plaintext* 1 : 123
- *Plaintext* 2 :456
- *key* 1 :12
- *key* 2 :34

➢ Menghitung *key schedule*

Proses pertama kali yang dilakukan SPECK adalah menghitung *key schedule*. Proses ini akan diulang sebanyak 32 kali.

Menghitung fungsi *round* x = 18 , y = 52 , k =0

- Menghitung X :
  - Mengkonversi nilai 18 menjadi biner :
    0000000000000000000000000000000000000000000000000000000000010010
  - Rotasi bit ke kanan sebanyak 8 kali :
    0101011000000000000000000000000000000000000000000000000000000000
  - X OR Y
    Hasil = 1001000000000000000000000000000000000000000000000000000000000110100
  - Nilai X tidak boleh diatas 18446744073709551615
  - X XOR k
    Hasil = 1001000000000000000000000000000000000000000000000000000000000110100

- Menghitung Y :
  - Mengkonversi nilai 52 menjadi biner :
    0000000000000000000000000000000000000000000000000000000000110100
  - Rotasi bit ke kiri sebanyak 3 kali :
    0000000000000000000000000000000000000000000000000000000110100000
  - Y XOR X
    Hasil = 1001000000000000000000000000000000000000000000000000000110010100

  Hasil X :1297036692682702900
  Hasil Y :1297036692682703252

➢ Melakukan enkripsi data

Proses kedua yang dilakukan SPECK adalah enkripsi data yang diulang sebanyak 32 kali.

----- Round 1 -----

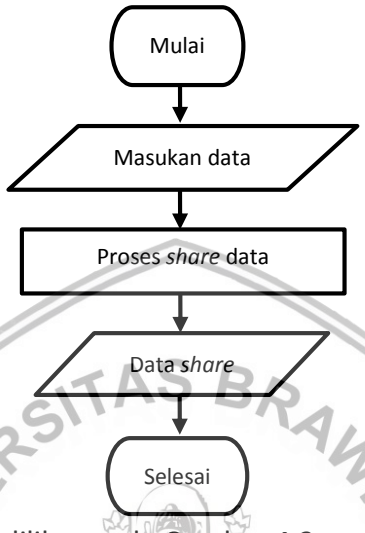
Menghitung fungsi *round* x = 291 , y = 1110 , k =52

- Menghitung X :
  - Mengkonversi nilai 291 menjadi biner :
    0000000000000000000000000000000000000000000000000100100011
  - Rotasi bit ke kanan sebanyak 8 kali :
    0010001100000000000000000000000000000000000000000000000000000001
  - X OR Y
    Hasil = 1000110000000000000000000000000000000000000000000010001010111
  - Nilai X tidak boleh diatas 18446744073709551615
  - X XOR k
    Hasil = 1000110000000000000000000000000000000000000000000010001100011
- Menghitung Y :
  - Mengkonversi nilai 1110 menjadi biner :
    0000000000000000000000000000000000000000000000000000010001010110
  - Rotasi bit ke kiri sebanyak 3 kali :
    0000000000000000000000000000000000000000000000000000000110100000
  - Y XOR X
    Hasil = 0000000000000000000000000000000000000000000000000010001010110000

  Hasil X :2522015791327478883
  Hasil Y :2522015791327487699

4.1.2 Algoritme Shamir's *Secret Sharing*

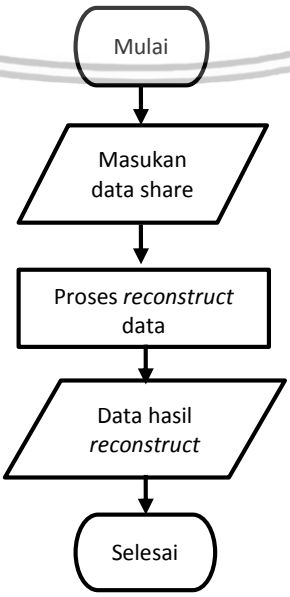
Berikut merupakan perancangan proses *share* dalam bentuk *flowchart* yang digunakan dalam sistem ini. Pertama pengguna memasukan data, setelah itu data akan dipecah sesuai dengan jumlah *share*. Data awal tadi akan menjadi data *share*. Proses memecah data ini menggunakan algoritme Shamir's *Secret Sharing*.



Bagan perancangan dapat dilihat pada Gambar 4.3.

Gambar 4.3 Proses *Share* Data

Setelah proses *share* terdapat juga proses *reconstruct*. Berikut merupakan perancangan proses *reconstruct* dalam bentuk *flowchart* yang digunakan dalam sistem ini. Pertama pengguna memasukan data *share* yang sudah dipecah. Setelah itu data akan melewati proses *reconstruct* sesuai dengan jumlah *share*, sehingga data *share* kembali menjadi data awal. Proses *reconstruct* data ini menggunakan algoritme Shamir's *Secret Sharing*. Bagan perancangan dapat dilihat pada Gambar 4.4.



Gambar 4.4 Proses *Reconstruct*

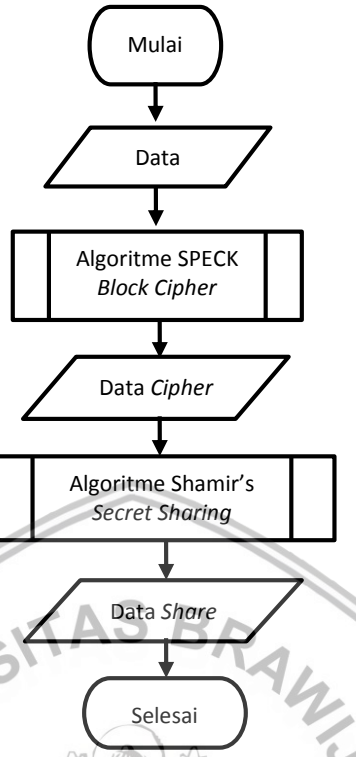
Dari alur sistem *share* dan *reconstruct* dapat di manualisasikan seperti langkah-langkah dibawah ini :

1. Untuk yang pertama adalah merubah biner dari *ciphertext* yang di peroleh dari proses enkripsi menjadi desimal untuk biner adalah: "11111000010110010110010111111011111000010001111001111001001010101101100010000100101011001110010100101000100110110110111000001110". Setelah itu di ubah menjadi desimal seperti berikut "330112725854378463219245462545273417230".
2. Setelah itu sistem akan membangun bilangan prima secara acak dengan batasan harus lebih besar dari bilangan desimal *ciphertext* "4949303300757142399006846970167897135 99" yang nantinya digunakan untuk proses *share*.
3. Kemudian menginputkan jumlah data yang ingin di pecah (*share*) dan jumlah data yang ingin di kembalikan (*reconstruct*). Setelah itu desimal *ciphertext* tersebut akan dipecah menjadi 3 bagian. Sehingga akan dihasilkan 3 macam bilangan desimal seperti berikut :
   - 2812887551810569116435885362897361414 20
   - 2324647845077353600679316100341988656 10
   - 1836408138344138084922746837786615898 00
4. Dalam proses *reconstruct* sistem akan membangun koefisien yang jumlahnya bilangan *reconstruct* dikurangi 1. Sehingga jika data yang akan di kembalikan adalah 3, maka koefisienya adalah 2. Kemudian untuk menggabungkan kembali *file* awal atau bilangan desimal awal, hanya diperlukan 2 dari 3 bagian file tersebut. Berikut adalah angka yang digunakan untuk *reconstruct* data :
   - 2812887551810569116435885362897361414 20
   - 2324647845077353600679316100341988656 10
5. Dari gabungan 2 pecahan ini akan menghasilkan bilangan desimal yang sesuai dengan bilangan desimal ciphertext yakni : "330112725854378463219245462545273417230".

4.2 Perancangan Sistem

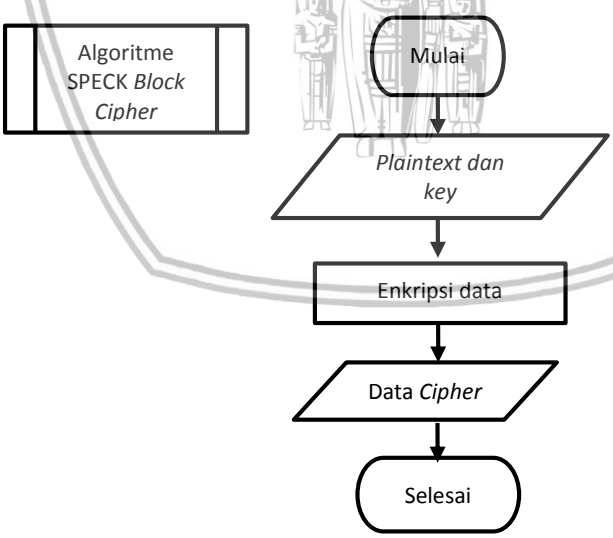
4.2.1 Proses Enkripsi dan *Share* Data

Berikut merupakan *flowchart* perancangan sistem berupa enkripsi dan *share* data yang digunakan untuk menjelaskan bagaimana data tersebut akan diproses dalam sistem ini. Pertama pengguna memasukan data berupa *file* teks, setelah itu akan melalui proses enkripsi dengan menggunakan algoritme SPECK *Block Cipher*. Kemudian terdapat data *cipher* yang sudah melalui proses enkripsi akan dipecah dengan menggunakan algoritme Shamir's *Secret Sharing*. Gambaran sistem dapat dilihat pada Gambar 4.5.

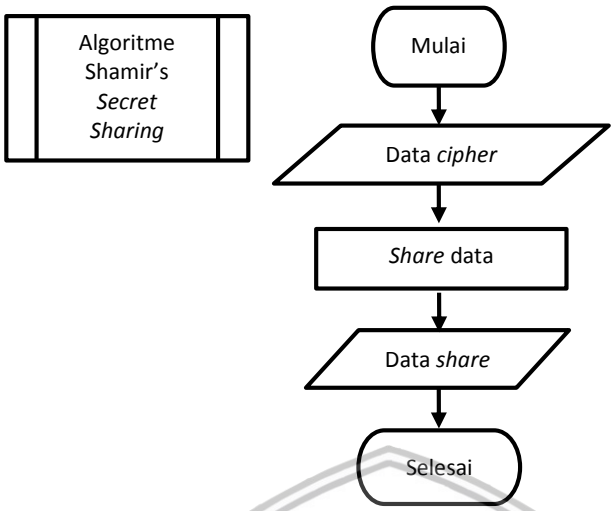


Gambar 4.5 Proses Enkripsi dan *Share* Data

Untuk proses enkripsi dari algoritme SPECK *block cipher* dapat dilihat pada Gambar 4.6, sedangkan untuk proses Shamir's *secret sharing* bisa dilihat pada Gambar 4.7.



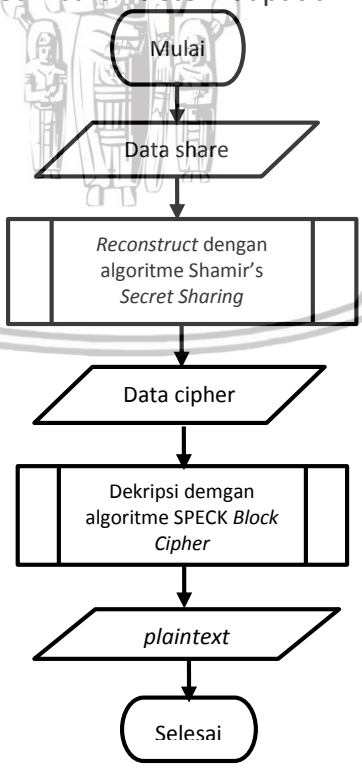
Gambar 4.6 Proses Enkripsi data



Gambar 4.7 Proses *Share* Data

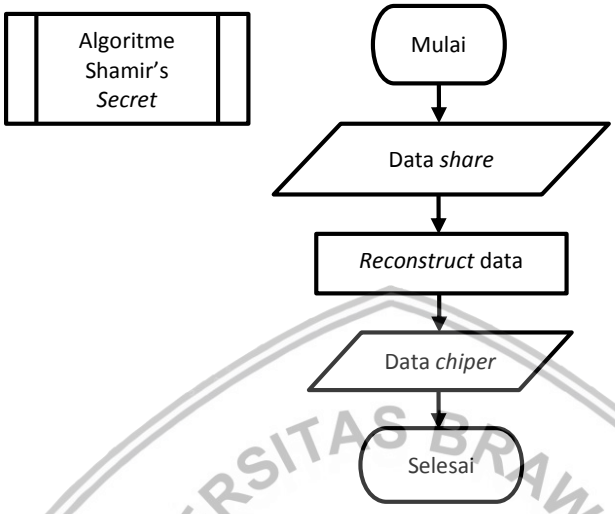
4.2.2 Proses *Reconstruct* dan Dekripsi Data

Berikut merupakan *flowchart* perancangan sistem berupa *reconstruct* dan dekripsi data yang digunakan untuk menjelaskan bagaimana data tersebut akan diproses dalam sistem ini. Pertama pengguna memasukan data yang sudah dipecah berupa *file* txt. Setelah itu akan melalui proses *reconstruct* dengan menggunakan algoritme Shamir's *Secret Sharing*, kemudian akan terdapat data *cipher* awal. Setelah itu data *cipher* akan di dekripsi dengan menggunakan algoritme SPECK *Block Cipher*. Gambaran sistem dapat dilihat pada Gambar 4.8.

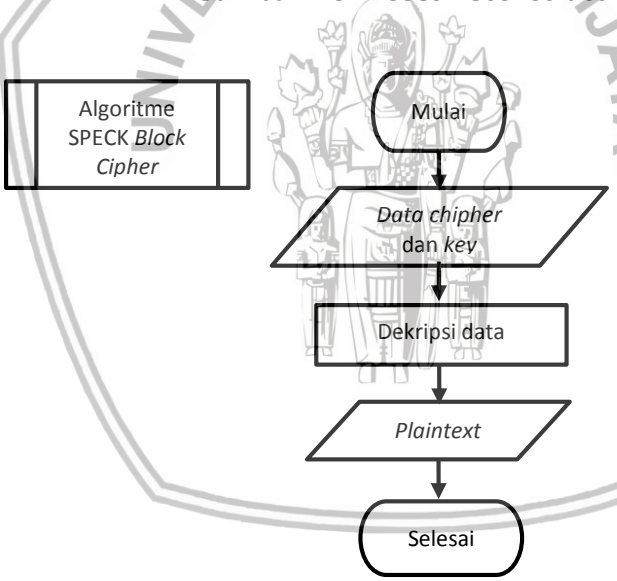


Gambar 4.8 Proses Dekripsi Data

Untuk proses Shamir's *secret sharing* bisa dilihat pada Gambar 4.9 , sedangkan untuk proses dekripsi dari algoritme SPECK *block cipher* dapat dilihat pada Gambar 4.10.



Gambar 4.9 Proses *Reconstruct*

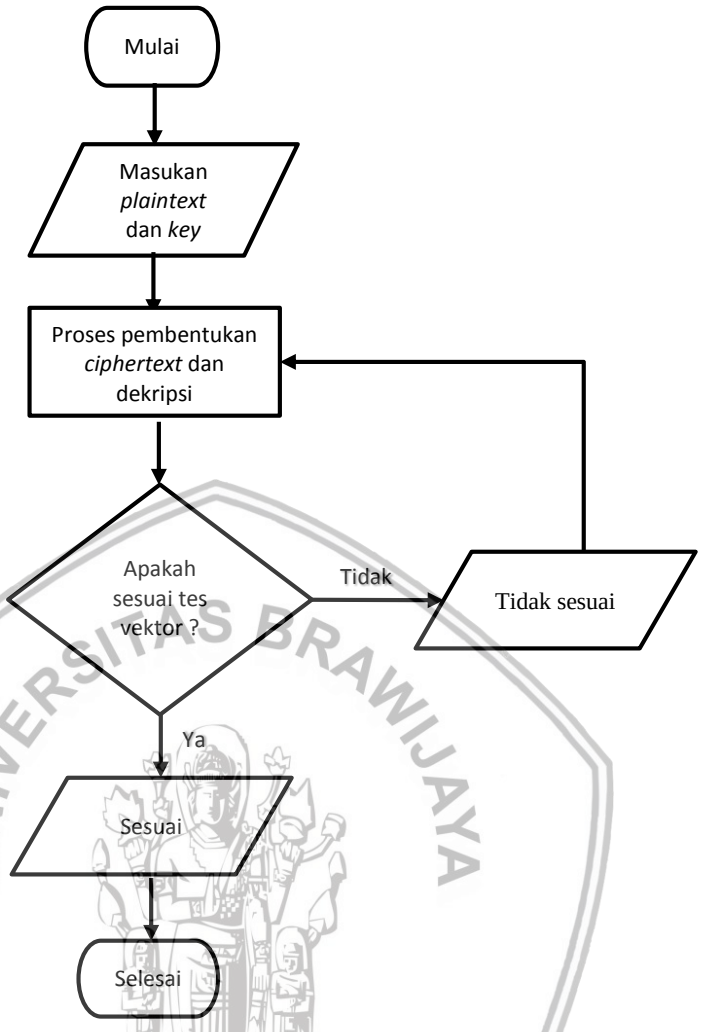


Gambar 4.10 Proses Dekripsi

4.3 Perancangan Pengujian

4.3.1 Pengujian *Test Vector*

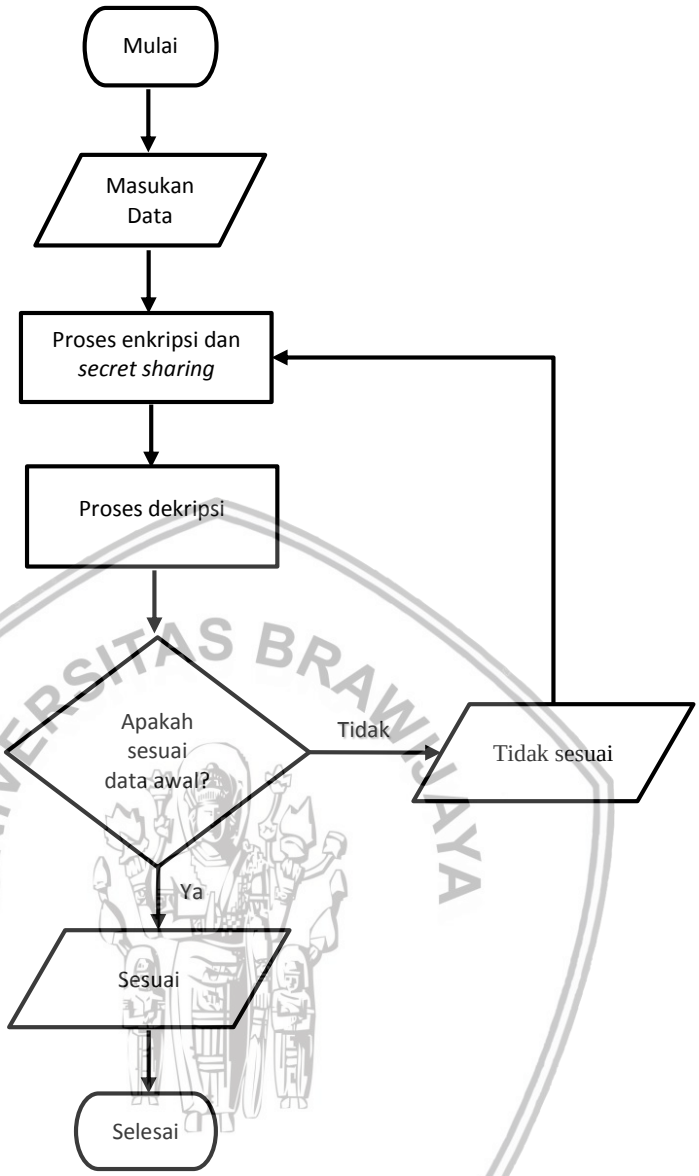
Pengujian ini dilakukan dengan tujuan untuk mengetahui bahwa sistem yang dibuat dengan menggunakan algortime SPECK *Block Cipher* sudah sesuai dengan *test vector* yang ada pada *paper* SPECK *Block Cipher* yang dibuat oleh Ray Beaulieu. Proses pengujian ini dimulai dari pengguna memasukkan *plaintext* dan *key*, kemudian akan diproses dengan algoritme SPECK *Block Cipher*. Gambar pengujian test vektor dapat dilihat pada Gambar 4.11.



Gambar 4.11 Pengujian Test Vektor

4.3.2 Pengujian Enkripsi dan Dekripsi

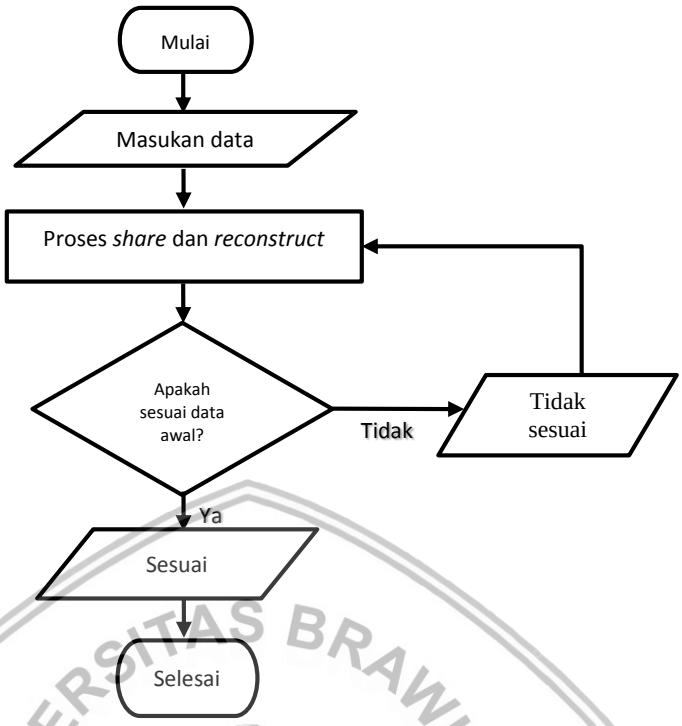
Pengujian ini dilakukan dengan tujuan untuk mengetahui bahwa algoritme yang menggunakan teknik enkripsi dan dekripsi berjalan dengan baik. Pengujian dengan teknik enkripsi dianggap berhasil jika data yang diamankan berubah menjadi data acak yang sulit dibaca. Pengujian dekripsi dapat dianggap berhasil jika data acak dapat kembali menjadi data semula. Proses pengujian ini dimulai dari pengguna memasukkan data, kemudian akan diproses dengan teknik enkripsi dan *secret sharing*. Setelah data melewati proses *secret sharing,* maka data akan memasuki proses dekripsi. Kemudian akan dicocokkan apakah hasilnya sama dengan data awal. Bagan dari pengujian ini dapat dilihat pada Gambar 4.12.



Gambar 4.12 Pengujian Enkripsi dan Dekripsi

4.3.3 Pengujian *Share* dan *Reconstruct* Data

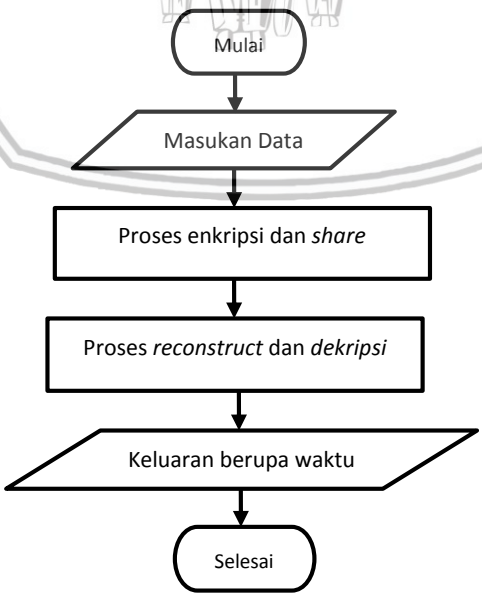
Pengujian ini dilakukan dengan tujuan untuk mengetahui bahwa sistem yang dibuat dengan algortime Shamir's *secret Sharing* berjalan dengan baik. Pengujian *share* dianggap berhasil apabila data yang dipecah dapat dikembalikan sesuai dengan data awal dengan jumlah *share* dan *reconstruct* yang berbeda-beda. Proses pengujian ini dimulai dari pengguna memasukkan data, setelah itu masuk proses *share* dan *reconstruct*. Apabila data sesuai dengan data awal maka sistem akan berhenti. Bagan pada pengujian ini dapat dilihat pada Gambar 4.13.



Gambar 4.13 Pengujian *Share* dan Reconstruct Data

4.3.4 Pengujian Waktu

Pengujian ini dilakukan dengan tujuan untuk mengetahui waktu yang dibutuhkan sistem untuk proses enkripsi, *share*, *reconstruct,* dan dekripsi pada *file* teks. Pengujian ini diawali dari pengguna memasukkan data, setelah itu data akan melewati proses enkripsi dan *share*. Kemudian melewati proses *reconstruct* dan dekripsi, setelah itu menampilkan output *file* beserta waktu yang dibutuhkan selama proses tersebut. Bagan pengujian ini dapat dilihat pada Gambar 4.14.



Gambar 4.14 Pengujian Waktu

BAB 5 IMPLEMENTASI

Bab lima menjelaskan tentang peranan implementasi sistem yang akan dilakukan dari tahapan setelah perancangan. Bab ini menjelaskan tentang implementasi sistem yang sudah di terapkan dalam kode program yakni mulai dari algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing*.

5.1 Algoritme SPECK *Block Cipher*

5.1.1 *Source Code Rotate Left*

Pada implementasi SPECK 128/128 terdapat *source code rotateLeft*. *Source code* ini berfungsi untuk memutar biner ke arah kiri. Pada tahap ini menggunakan inputan angka dan jumlah. Potongan kode program dapat dilihat pada *source code* 1.

Source Code 1: Fungsi Rotate Left

```
1   public static BigInteger rotateLeft(BigInteger bil, int jum) {
2           String str = bil.toString(2);
3           char[] binerFNumb = str.toCharArray();
4           char[] biner = new char[64];
5           int i,j;
6           for(j=biner.length-1; j>=0; j--){
7               if(biner.length-binerFNumb.length <= j)
8                   biner[j]    =    binerFNumb[binerFNumb.length    -
9   (biner.length-j)];
10              else
11                  biner[j] = '0';
12          }
13          char temp;
14          for(i=1; i<=jum; i++){
15              temp = biner[0];
16              for(j=0; j<=biner.length-2; j++){
17                  biner[j] = biner[j+1];
18              }
19              biner[biner.length-1] = temp;
20          }
21
22          str = String.valueOf(biner);
23          BigInteger result = new BigInteger(str, 2);
24          return result;
25      }
```

5.1.2 *Source Code Rotate Right*

Pada implementasi SPECK 128/128 terdapat *source code rotateRight*. *Source code* ini berfungsi untuk memutar biner ke arah kanan. Pada tahap ini menggunakan inputan angka dan jumlah. Potongan kode program dapat dilihat pada *source code* 2.

Source Code 2 : Fungsi Rotate Left

```
1   public static BigInteger rotateRight(BigInteger bil, int jum) {
2           String str = bil.toString(2);
3           char[] binerFNumb = str.toCharArray();
4           char[] biner = new char[64];
```

Source Code 2 : Fungsi Rotate Left (lanjutan)

```
5           int i,j;
6           for(j=biner.length-1; j>=0; j--){
7               if(biner.length-binerFNumb.length <= j)
8                   biner[j]   =   binerFNumb[binerFNumb.length   -
9   (biner.length-j)];
10              else
11                  biner[j] = '0';
12          }
13
14          char temp;
15          for(i=1; i<=jum; i++){
16              temp = biner[biner.length-1];
17              for(j=biner.length-1; j>=1; j--){
18                  biner[j] = biner[j-1];
19              }
20              biner[0] = temp;
21          }
22
23          str = String.valueOf(biner);
24          BigInteger result = new BigInteger(str, 2);
25          return result;
26      }
```

5.1.3 *Source Code* SPECK *Round*

Pada implementasi SPECK 128/128 terdapat *source code* SPECK *round. Source code* ini digunakan untuk membuat *round* pada algoritme SPECK. Fungsi *round* ini di jalankan sesuai dengan tipe SPECK. Apabila SPECK menggunakan *block* 128 maka *round* yang di jalankan sebanyak 32 kali. Potongan kode program dapat dilihat pada *source code* 3.

Source Code 3 : Fungsi SPECK Round

```
1   public static BigInteger[] speckRound(BigInteger x, BigInteger y,
2   BigInteger k){
3           x = rotateRight(x,alpha);
4           x = x.add(y);
5
6           BigInteger lim = new BigInteger(addmod);
7           BigInteger limPlus1 = new BigInteger(addmod1);
8           while (x.compareTo(lim) > 0){
9               x = x.subtract(limPlus1);
10          }
11          x = x.xor(k);
12          y = rotateLeft(y, beta);
13          y = y.xor(x);
14          BigInteger[] result = new BigInteger[2];
15          result[0] = x;
16          result[1] = y;
17          return result;
18      }
```

5.1.4 *Source Code* SPECK *Reversed Round*

Pada implementasi SPECK 128 terdapat *source code reversed round*. *Source code* ini berfungsi untuk mengembalikan SPECK yang sudah di *rotate*. Pada tahap ini menggunakan inputan x, y dan k dengan tipe BigInteger. Potongan kode program dapat dilihat pada *source code* 4.

Source Code 4 : Fungsi SPECK Reversed Round

```
1   public static BigInteger[] speckReverseRound(BigInteger x,
2   BigInteger y, BigInteger k){
3
4           y = x.xor(y);
5           y = rotateRight(y, beta);
6
7           x = x.xor(k);
8           x = x.subtract(y);
9
10          BigInteger zero = new BigInteger("0");
11          BigInteger limPlus1 = new BigInteger(addmod1);
12          while (x.compareTo(zero) < 0){
13              x = x.add(limPlus1);
14          }
15
16          x = rotateLeft(x, alpha);
17
18          BigInteger[] result = new BigInteger[2];
19          result[0] = x;
20          result[1] = y;
21          return result;
22      }
```

5.1.5 *Source Code Key Scheduling*

Pada implementasi SPECK 128/128 terdapat proses *key scheduling* yang digunakan untuk membuat kunci yang akan digunakan dalam proses enkripsi dan dekripsi algoritme SPECK. Potongan kode program dapat dilihat pada *source code* 5.

Source Code 5 : Fungsi Key Scheduling

```
1   public static BigInteger[] keyScheduling(BigInteger[] key){
2           BigInteger[] result = new BigInteger[T];
3           BigInteger[] temp = new BigInteger[2];
4           BigInteger k1 = key[0];
5           BigInteger k2 = key[1];
6           result[0] = k2;
7       for (int i = 0; i < T-1; i++) {
8               String str = String.valueOf(i);
9               BigInteger j = new BigInteger(str);
10              temp = speckRound(k1, k2, j);
11              k1 = temp[0];
12              k2 = temp[1];
13              result[i+1] = k2;
14
15          }
16          return result;
17      }
```

5.1.6 *Source Code* Enkripsi

Pada implementasi SPECK 128/128 terdapat proses enkripsi yang digunakan untuk mengubah *plaintext* menjadi *ciphertext*. Proses ini terdapat pada fungsi Encryption().Potongan kode program dapat dilihat pada *source code* 6.

Source Code 6 : Fungsi Encryption

```
1  public static BigInteger[] encryption(BigInteger[] plaintext,
2  BigInteger[] keySchedule) throws IOException{
3
4          BigInteger[] ciphertext = new BigInteger[2];
5          ciphertext[0] = plaintext[0];
6          ciphertext[1] = plaintext[1];
7          for(int i=0; i<= T-1; i++){
8              ciphertext = speckRound(ciphertext[0], ciphertext[1],
9  keySchedule[i]);
10         }
11
12         return ciphertext;
13
14
15     }
```

5.1.7 *Source Code* Dekripsi

Pada implementasi SPECK 128 terdapat proses dekripsi yang digunakan untuk mengubah *ciphertext* menjadi *plaintext*. Proses ini terdapat pada fungsi decryption().Potongan kode program dapat dilihat pada *source code* 7.

Source Code 7 : Fungsi Decryption

```
1  public static BigInteger[] decryption(BigInteger[] ciphertext,
2  BigInteger[] keySchedule){
3
4          BigInteger[] plaintext = new BigInteger[2];
5          plaintext[0] = ciphertext[0];
6          plaintext[1] = ciphertext[1];
7
8          for(int i=T-1; i>=0; i--){
9              plaintext      =      speckReverseRound(plaintext[0],
10 plaintext[1], keySchedule[i]);
11         }
12
13         return plaintext;
14     }
```

5.1.8 *Source Code* Konversi *Byte to Word*

Terdapat fungsi convertBytetoWord() yang digunakan untuk mengubah *byte* menjadi setiap kata. Pada fungsi ini terdapat parameter dengan tipe *byte*. Potongan kode program dapat dilihat pada *source code* 8.

Source Code 8 : Fungsi Konversi Byte to Word

```
1  public static BigInteger [] convertBytetoWord(byte [] number){
2          String [] binaryNumber = new String [number.length];
3          BigInteger temp ;
4          String basis10;
5
```

Source Code 8 : Fungsi Konversi Byte to Word (lanjutan)

```
6         for (int i=0; i < number.length; i++){
7               basis10=Byte.toString(number[i]);
8               temp = new BigInteger(basis10);
9               binaryNumber [i] = temp.toString(2);//basis 2
10
11              int pad = 8 - binaryNumber[i].length();
12              for (int j=1;j<=pad;j++){
13                  binaryNumber[i]="0"+binaryNumber[i];
14              }
15          }
16
17          int pjgStr = number.length / 8 + 1;
18
19          boolean cek = false; //kalau false tidak ada pembulatan
20          if(pjgStr % 2 == 1){
21              cek = true;
22              pjgStr = pjgStr + 1;
23          }
24          int sisa=number.length;
25
26          int gabung = 8;
27          String a;
28          BigInteger [] hasil = new BigInteger[pjgStr];
29          for(int i=0; i<pjgStr; i++ ){
30              if((cek)&& (sisa <=0)){
31                  hasil[i]=new BigInteger("0");
32              }else{
33                  if(sisa>=8){
34                      gabung = 8;
34                      sisa = sisa - gabung;
35
36                  }else{
37                      gabung = sisa;
38                      sisa = sisa - gabung;
39
40                  }
41                  a = "";
42                  for(int j=i*8;j<i*8+gabung;j++){
43                      a = a+ binaryNumber[j];
44                  }
45                  hasil[i] = new BigInteger(a,2);
46                  //System.out.println(hasil[i]);
47
48              }
49          }
50
51           return hasil;
52      }
```

5.1.9 *Source Code* Konversi *Word to Byte*

Fungsi ini digunakan untuk mengubah *word* menjadi *byte*. Pada fungsi ini terdapat parameter dengan tipe BigInteger dan integer. Potongan kode program dapat dilihat pada *source code* 9.

Source Code 9 : Fungsi Konversi Word to Byte

```
1   public static byte [] convertWordtoByte(BigInteger [] data, int
2   byteLength){
3           byte [] result = new byte[(int) byteLength];
4           String [] biner = new String [data.length];
5           String str;
6           int i,j,k;
7           for(i=0; i<data.length;i++){
8               biner [i]=data[i].toString(2);
9           }
10          int remain, gabung = 8;
11          remain = byteLength;
12
13          i=0;
14          while(remain>0){//untuk perulangan bitlength
15              if(remain>=8){
16                  gabung=8;
17                  remain=remain-gabung;
18              }else{
19                  gabung=remain;
20                  remain=remain-gabung;
21              }
22              k = gabung * 8 - biner[i].length();
23              for (j = 1;j<=k;j++){
24                  biner[i]="0"+biner[i];
25              }
26              k = 0;
27
28              for (j=i*8; j<i*8+gabung;j++){
29                  if(j<i*8+gabung-1){
30                      str = biner[i].substring(k, k+8);
31                  }else{
32                      str = biner[i].substring(k);
33                  }
34                  k=k+8;
34                  result[j]=Byte.valueOf(str, 2);
35              }
36              i=i+1;
37          }
38          return result;
39      }
```

5.2 Algoritme Shamir's *Secret Sharing*

5.2.1 *Source Code Share*

Pada algoritme Shamir's *Secret Sharing* terdapat *source code share* yang berfungsi sebagai pemecah *secret key* yang diinginkan pengguna dan menghasilkan *share key*. Terdapat beberapa inputan yang dibutuhkan yaitu nilai *secret, needed, available,* prima dan random. Tujuan dari proses *share* itu sendiri adalah untuk meningkatkan *availability secret key* tersebut. Misalnya, jika ada *share key* yang hilang, maka pengguna akan mudah untuk melakukan *reconstruct secret key* awal dengan *share* yang tersisa. Potongan kode program dapat dilihat pada *source code* 9.

Source Code 9 : Fungsi *Share*

```
1     public static SecretSharing [] splitting(BigInteger secret, int
2  bth, int available, BigInteger prima,Random random) throws
3  IOException
4      {
5          long start4 = System.currentTimeMillis();
6          final       BigInteger[]       koefisien       =       new
7  BigInteger[bth];//menentukan koefisien
8          koefisien[0] = secret;
9          for (int i = 1; i < bth; i++)
10         {
11             BigInteger r;
12             while (true)
13             {
14                 r = new BigInteger(prima.bitLength(), random);
15                 if    (r.compareTo(BigInteger.ZERO)    >    0    &&
16 r.compareTo(prima) < 0)
17                 {
18                     break;
19                 }
20             }
21             koefisien[i] = r;
22         }
23
24         final       SecretSharing[]       share       =       new
25 SecretSharing[available];
26         for (int x = 1; x <= available; x++)
27         {
28             BigInteger akumulasi = secret;
29             for (int exp = 1; exp < bth; exp++) {
30                 akumulasi                                          =
31 akumulasi.add(koefisien[exp].multiply(BigInteger.valueOf(x).pow(ex
32 p).mod(prima))).mod(prima);
33             }
34             share[x - 1] = new SecretSharing(x, akumulasi);
34             String str = akumulasi.toString();
35             File  newpdf  =  new  File("D:/file/hasil/share  ke-"  +
36 x+".txt");
37             FileWriter fwpdf = new FileWriter(newpdf);
38             fwpdf.write(str);
39             fwpdf.close();
40             System.out.println(share[x - 1]);
41
42             System.out.println("----------------------------------
43 ----------------------------------");
44           }
45         long end4 = System.currentTimeMillis();
46         System.out.println("\nWaktu    yang    diperlukan    dalam
47 menghasilkan  split  file  adalah  "  +  formatter1.format((end4  -
48 start4) / 1000.0) + " seconds");
49         return share;
       }
```

5.2.2 *Source Code Reconstruct*

Source code reconstruct digunakan untuk mengembalikan *file* yang sudah dipecah, proses ini terdapat pada fungsi combine(). Potongan kode program dapat dilihat pada *source code* 10.

Source Code 10 : Fungsi Reconstruct

```
1   public static BigInteger combine(final SecretSharing[] share,
2   final BigInteger prima)
3       {
4           long start3 = System.currentTimeMillis();
5           BigInteger akumulasi = BigInteger.ZERO;
6
7           for(int formula = 0; formula < share.length; formula++)
8           {
9               BigInteger pembilang = BigInteger.ONE;
10              BigInteger penyebut = BigInteger.ONE;
11
12              for(int count = 0; count < share.length; count++)
13              {
14                  if(formula == count)
15                      continue; // If not the same value
16
17                  int posawal = share[formula].getNumber();
18                  int nextpos = share[count].getNumber();
19
20                  pembilang                                   =
21  pembilang.multiply(BigInteger.valueOf(nextpos).negate()).mod(prima
22  ); // (pembilang * -nextpos) % prima;
23                  penyebut                                    =
24  penyebut.multiply(BigInteger.valueOf(posawal                -
25  nextpos)).mod(prima); // (penyebut * (posawal - nextpos)) % prima;
26              }
27              BigInteger value = share[formula].getShare();
28              BigInteger   tmp    =   value.multiply(pembilang)    .
29  multiply(modInverse(penyebut, prima));
30              akumulasi = prima.add(akumulasi).add(tmp).mod(prima);
31  //      (prima   +   akumulasi   +   (value   *   pembilang   *
32  modInverse(penyebut))) % prima;
33          }
34          long end3 = System.currentTimeMillis();
34          System.out.println("==================================
35  HASIL ==================================");
36          System.out.println("Result Reconstruct : " + akumulasi);
37          System.out.println("Result  Reconstruct  biner  :  "  +
38  akumulasi.toString(2));
39          System.out.println("\nWaktu   yang   diperlukan   dalam
40  menghasilkan reconstruct adalah " + formatter1.format((end3 -
41  start3) / 1000.0) + " seconds");
42
43          return akumulasi;
44      }
```

5.3 *Source Code* Konversi

Source code konversi bertujuan untuk mengkonversi suatu bilangan. Terdapat beberapa konversi seperti konversi string ke biner, konversi biner ke ASCII, konversi biner ke desimal.

5.3.1 *Source Code* Konversi *String* ke Biner

Source code konversi *string* ke biner digunakan untuk mengkonversi bentuk *string* menjadi bentuk biner. Konversi ini digunakan setiap 8 bit. Potongan kode program dapat dilihat pada *source code* 13.

Source Code 13 : Fungsi Convert String Biner

```
1   public static String convertStringBiner(String kata) {
2           String result = "";
3           String tempStr;
4           int tempInt;
5           char[] StrChar = kata.toCharArray();
6           for (int i = 0; i < StrChar.length; i++) {
7               tempStr = Integer.toBinaryString(StrChar[i]);
8
9               tempInt = tempStr.length();
10              if (tempInt != 8) {
11                  tempInt = 8 - tempInt;
12                  if (tempInt == 8) {
13                      result += tempStr;
14                  } else if (tempInt > 0) {
15                      for (int j = 0; j < tempInt; j++) {
16                          result += "0";
17                      }
18                  } else {
19                      System.out.println("too long...");
20                  }
21              }
22              result += tempStr;
23          }
24          result += "";
25          return result;
26      }
```

5.3.2 *Source Code* Konversi Biner ke ASCII

Source code konversi biner ke ASCII digunakan untuk mengkonversi bentuk biner menjadi bentuk ASCII. Konversi ini digunakan setiap 8 bit. Potongan kode program dapat dilihat pada *source code* 14.

Source Code 14 : Fungsi Convert Biner To ASCII

```
1   public static String convertBintoAscii(String biner) {
2           String inputbin = biner;
3           String outputbin = "";
4           for (int i = 0; i <= inputbin.length() - 8; i += 8) {
5               int k = Integer.parseInt(inputbin.substring(i, i + 8),
6   2);
7               outputbin += (char) k;
8           }
9           return outputbin;
10      }
```

5.3.3 *Source Code* Konversi Biner ke Desimal

Source code konversi biner ke desimal digunakan untuk mengkonversi bentuk biner menjadi bentuk desimal. Potongan kode program dapat dilihat pada *source code* 15.

Source Code 15 : Fungsi Konversi Biner To Decimal

```
1   public static BigDecimal BInerToDecimal(String binStr) {
2
3           BigDecimal jum = new BigDecimal("0");
4           BigDecimal basis = new BigDecimal(2);
5           BigDecimal temp;
6           for (int i = 0; i < binStr.length(); i++) {
7               if (binStr.charAt(i) == '1') {
8                   int exp = binStr.length() - 1 - i;
9                   temp = basis.pow(exp);
10                  jum = jum.add(temp);
11              }
12
13          }
14          return jum;
15      }
```

5.4 *Source Code* Pengambilan *File* Dari Komputer

Source code pengambilan *file* dari komputer digunakan untuk mengambil *file* yang sudah dipecah oleh sistem, yang digunakan untuk merekonstruksi pecahan *file* sehingga menjadi satu bagian. Potongan kode program dapat dilihat pada *source code* 16.

Source Code 16 : Pengambilan file dari komputer

```
1   for (int i = 1; i <= rct; i++) {
2                               System.out.print("Input     No     File
3   Reconstruct : ");
4                               no = in.nextInt();
5                               for (int j = 0; j < sh; j++) {
6                                   String          read          =
7   readFile("D:/file/hasil/share ke-" + no + ".txt");
8                                   BigInteger   akumulasi   =   new
9   BigInteger(read);
10                                  shares2[j] = new SecretSharing(no,
11  akumulasi);
12                                  //System.out.println("Read
13  :"+read);
14                              }
15                              shareSecret[i-1] = shares[no-1];
16                          }
```

BAB 6 PENGUJIAN DAN PEMBAHASAN

6.1 Pengujian Tes Vektor

Pengujian tes vektor merupakan pengujian yang digunakan untuk menguji kevalidan *file* pada suatu algoritme tersebut. Pada algoritme SPECK untuk menguji tes vektor terdapat tiga parameter yaitu *plaintext, key* dan *ciphertext* dengan tipe heksadesimal. Gambar dari tes vektor SPECK 128/128 dari *paper* asli dapat dilihat pada Gambar 6.1.

```
SPECK 128/128
Key:        0f0e0d0c0b0a0908 0706050403020100
Plaintext:  6c61766975716520 7469206564616d20
Ciphertext: a65d985179783265 7860fedf5c570d18
```

Gambar 6.1 Tes vektor pada jurnal SPECK 128/128

Hasil implementasi program membuktikan bahwa tes vektor SPECK 128/128 sama dengan tes vektor yang ada di jurnal SPECK asli. Hal ini membuktikan bahwa implementasi SPECK pada program penulis benar. Hasil tes vektor pada implementasi SPECK pada jurnal ini terdapat pada Gambar 6.2.

```
Input plaintext 1 :6c61766975716520
Input plaintext 2 :7469206564616d20

Input key 1 :0f0e0d0c0b0a0908
Input key 2 :0706050403020100

chiphertext 1 = a65d985179783265
ciphertext 2  = 7860fedf5c570d18
plaintext 1 = 6c61766975716520
plaintext 2 = 7469206564616d20
```

Gambar 6.2 Hasil pengujian tes vektor

6.2 Pengujian *Avalanche Effect*

Pengujian *avalanche effect* bertujuan untuk mengukur tingkat keamanan suatu algoritme kriptografi. Cara menguji *avalanche effect* yaitu dengan merubah satu bit pada *plaintext* atau *key* kemudian dibandingkan dengan masukan *plaintext* atau *key* yang mengalami perubahan 1 digit.

6.2.1 Perbandingan *Plaintext*

Berikut merupakan perbandingan *avalanche effect* pada *input plaintext* menggunakan algoritme SPECK *block cipher* varian 128/128 dengan dua kali percobaan yang bisa dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian *Avalanche Effect* Perbandingan *Plaintext*

Pengujian Pertama	
Plaintext 1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Plaintext 2	10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Chiphertext 1	10111100 10110101 11001000 10001110 01111010 01011011 11100101 01100110
Chiphertext 2	01000010 00111011 01010001 11010001 10110001 01100101 01101011 10001000
Pengujian Kedua	
Plaintext 1	10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Plaintext 2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Chiphertext 1	00101111 10110110 10000000 10111110 10101001 10010001 01101110 10101110
Chiphertext 2	01101010 11001110 00000100 01010110 10011001 01011011 01001111 10111100

Pada Tabel 6.1 menunjukkan hasil dari pengujian *avalanche effect* pada algoritme SPECK *block cipher* varian 128/128 seperti berikut:

- Pada pengujian pertama hasil dari pengujian *avalanche effect* adalah 0,52.
- Pada pengujian kedua hasil dari pengujian *avalanche effect* adalah 0,53.

Jadi rata-rata dari dua pengujian tersebut adalah 0,53, yang berarti nilai *avalanche effect* algoritme SPECK *block cipher* varian 128/128 adalah 53%. Hasil dari pengujian ini membuktikan bahwa algoritme SPECK 128/128 terbukti keamanannya.

6.2.2 Perbandingan *Key Schedule*

Berikut merupakan perbandingan *avalanche effect* pada *input plaintext* menggunakan algoritme SPECK *block cipher* dengan dua kali percobaan yang bisa dilihat pada Tabel 6.2.

Tabel 6.2 Pengujian *Avalanche Effect* Perbandingan *Key Schedule*

Pengujian Pertama	
Plaintext 1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Plaintext 2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 2	10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Chiphertext 1	00011100 01100111 11110011 10100111 00100011 00100001 10011011
Chiphertext 2	01101111 01011001 11101101 11111101 00111000 00010100 01100001 00100111
Pengujian Kedua	
Plaintext 1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Plaintext 2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 1	10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Key Schedule 2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Chiphertext 1	01000111 5 10100001 3 10100111 5 01001000 2 10110010 4 10010111 5 00011110 4 11100111 6
Chiphertext 2	10011010 4 01010100 3 11010101 5 11110010 5 00110100 3 11111111 8 10010010 3 11100010 4

Pada Tabel 6.2 menunjukkan hasil dari pengujian *avalanche effect* pada algoritme SPECK *block cipher* varian 128/128 seperti berikut :

- Pada pengujian pertama hasil dari pengujian *avalanche effect* adalah 0,51.
- Pada pengujian kedua hasil dari pengujian *avalanche effect* adalah 0,54.

Jadi rata-rata dari dua pengujian tersebut adalah 0,53, yang berarti nilai *avalanche effect* algoritme SPECK *block cipher* varian 128/128 adalah 53%. Hasil dari pengujian ini membuktikan bahwa algoritme SPECK 128/128 terbukti keamanannya.

6.3 Pengujian Fungsional

Pengujian fungsional bertujuan untuk menguji apakah sistem sudah berjalan dengan benar. Terdapat beberapa pengujian fungsional yang dilakukan antara lain pengujian *input plaintext*, *generate key*, enkripsi, *share*, *reconstruct*, dekripsi, dan pengambilan *file* dari komputer.

6.3.1 Pengujian Fungsional *Input Plaintext*

Pengujian fungsional *input plaintext* bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan *input plaintext* yang sudah diatur pada program sehingga bisa berjalan sesuai prosedur yang sudah direncanakan. Pengujian fungsional *input plaintext* bisa dilihat pada Tabel 6.3.

Tabel 6.3 Pengujian Fungsional *Input Plaintext*

Nama Kasus Uji	Input *Plaintext*
Prosedur	Sistem akan menginputkan *plaintext* secara manual yang terletak pada program.
Hasil yang diharapkan	Sistem menginputkan *plaintext* secara manual yang terletak pada program.
Hasil	Sistem menampilkan input dari isi *file* yang telah di atur program.
Status	Valid

6.3.2 Pengujian Fungsional Proses *Generate Key*

Pengujian fungsional proses *generate key* bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan *generate key* dengan adanya kombinasi angka dan huruf. Pengujian fungsional proses *generate key* bisa dilihat pada Tabel 6.4.

Tabel 6.4 Pengujian Fungsional Proses *Generate Key*

Nama Kasus Uji	Proses *generate key*
Prosedur	Sistem akan dapat memproses inputan *key* dengan kombinasi angka dan huruf.
Hasil yang diharapkan	Sistem dapat memproses inputan *key* dengan kombinasi angka dan huruf.
Hasil	Sistem dapat memproses inputan key dengan kombinasi angka dan huruf, sehingga dapat masuk pada proses selanjutnya.
Status	Valid

6.3.3 Pengujian Fungsional Proses Enkripsi

Pengujian fungsional proses enkripsi bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan enkripsi data sehingga data tidak bisa dibaca. Pengujian fungsional proses enkripsi bisa dilihat pada Tabel 6.5.

Tabel 6.5 Pengujian Fungsional Proses Enkripsi

Nama Kasus Uji	Proses enkripsi
Prosedur	Sistem akan menjalankan proses enkripsi sehingga menghasilkan *chipertext*.
Hasil yang diharapkan	Sistem menjalankan proses enkripsi sehingga menghasilkan *chipertext*.
Hasil	Sistem menampilkan *chiphertext*
Status	Valid

6.3.4 Pengujian Fungsional Proses *Share*

Pengujian fungsional proses *share* bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan *share file* dengan memecah *file* menjadi beberapa bagian yang sudah di tentukan. Pengujian fungsional proses *share* bisa dilihat pada Tabel 6.6.

Tabel 6.6 Pengujian Fungsional Proses *Share*

Nama Kasus Uji	Proses *share*
Prosedur	Sistem akan menjalankan proses *share* sehingga menghasilkan pecahan *file*.
Hasil yang diharapkan	Sistem menjalankan proses *share* sehingga menghasilkan pecahan *file*.
Hasil	Sistem menghasilkan *share file*
Status	Valid

6.3.5 Pengujian Fungsional Proses *Reconstruct*

Pengujian fungsional proses *reconstruct* bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan *reconstruct file* sehingga data yang dipecah bisa menjadi satu bagian. Pengujian fungsional *proses reconstruct* bisa dilihat pada Tabel 6.7.

Tabel 6.7 Pengujian Fungsional Proses *Reconstruct*

Nama Kasus Uji	Proses *reconstruct*
Prosedur	Sistem akan menjalankan proses *reconstruct* sehingga pecahan *file* menjadi satu.
Hasil yang diharapkan	Sistem menjalankan proses *reconstruct* sehingga pecahan *file* menjadi satu.
Hasil	Sistem menghasilkan *reconstruct file*
Status	Valid

6.3.6 Pengujian Fungsional Proses Dekripsi

Pengujian fungsional proses dekripsi bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan dekripsi *file* sehingga *file* yang tidak

bisa dibaca menjadi *file* yang bisa dibaca. Pengujian fungsional *input plaintext* bisa dilihat pada Tabel 6.8.

Tabel 6.8 Pengujian Fungsional Proses Dekripsi

Nama Kasus Uji	Proses dekripsi
Prosedur	Sistem akan memproses dekripsi pada *file* yang sudah dienkripsi sehingga dapat kembali menjadi isi *file* awal.
Hasil yang diharapkan	Sistem memproses dekripsi pada *file* yang sudah dienkripsi sehingga dapat kembali menjadi isi *file* awal.
Hasil	Sistem menghasilkan dekripsi *file* awal
Status	Valid

6.3.7 Pengujian Fungsional Pengambilan *File* Dari Komputer

Pengujian fungsional *pengambilan file* dari komputer bertujuan untuk mengetahui apakah program yang sudah dibuat bisa melakukan pengambilan *file* yang ada di komputer. Pengujian fungsional *input plaintext* bisa dilihat pada Tabel 6.9.

Tabel 6.9 Pengujian Proses Pengambilan *File* Dari Komputer

Nama Kasus Uji	Proses pengambilan *file* dalam komputer
Prosedur	Sistem akan mengambil *file* dari komputer
Hasil yang diharapkan	Sistem mengambil *file* dari komputer
Hasil	Sistem berhasil mengambil *file* dari komputer
Status	Valid

6.4 Pengujian Validitas Enkripsi dan Dekripsi

Pengujian enkripsi dan dekripsi dilakukan untuk mengetahui apakah *file* tersebut benar-benar dapat melewati proses enkripsi dan dekripsi dengan baik. Pada pengujian ini dilakukan dengan mengambil *text* yang ada di PDF dengan menggunakan *library* i-Text. Kunci yang digunakan adalah kunci yang sama yaitu “abcd1234”. Ukuran *file* yang digunakan adalah 4 KB dengan baris yang berbeda yaitu 3 baris, 8 baris dan 13 baris. Pengujian validitas enkripsi dan dekripsi dapat dilihat pada Tabel 6.10.

Tabel 6.10 Validitas Enkripsi dan Dekripsi

Ukuran *File* dan Isi *File*	***Plaintext***	**Enkripsi**	**Dekripsi**	**Status**
4 KB, 3 baris	Pada kemajuan teknologi saat ini, orang	¤>~_!bO¯¸G¶_k_V_-TÁàµSURtz`ð6÷MIý|{_ _%øt/._Ù+cõ{¼Äb-_±	Pada kemajuan teknologi saat	Valid

Tabel 6.10 Validitas Enkripsi dan Dekripsi (Lanjutan)

Ukuran *File* dan Isi *File*	*Plaintext*	Enkripsi	Dekripsi	Status
	Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF.	¤>~_!bO¯¸G¶_k_V_-T ÁàµSURtz`ð6÷MIý|{_ _%øt/._Ù+cõ{¼Äb-_±' Åë¶k+_*3Vþ_\Î¨J;Q{+ ôÄû#÷>- Ð_G_•3/Úh£ÛÐÐH_ Ò@ò_÷_ëf_glë_§vï_¥ PÛÜ¼÷àÎ«Äò<&D·[ý_P U_£_¼_Oýiõ ýÛ@S¨_U ùq$©þ_6üèCHÐJD È2~AF~YÄ_ fn§Í3e}ÌîÂG}mgæE_âT 8v;¯ÑCc¯ÊF³lÉJ¹jÿGPÓ _*¶»¼_E__ÆÉ«YKv	Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF.	
4 KB, 8 baris	Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal	¤>~_!bO¯¸G¶_k_V_-T ÁàµSURtz`ð6÷MIý|{_ _%øt/._Ù+cõ{¼Äb-_±' Åë¶k+_*3Vþ_\Î¨J;Q{+ ôÄû#÷>- Ð_G_•3/Úh£ÛÐÐH_ Ò@ò_÷_ëf_glë_§vï_¥ PÛÜ¼÷àÎ«Äò<&D·[ý_P U_£_¼_Oýiõ ýÛ@S¨_U ùq$©þ_6üèCHÐJD È2~AF~YÄ_ fn§Í3e}ÌîÂG}mgæE_âT 8v;¯ÑCc¯ÊF³lÉJ¹jÿGPÓ _*¶»ÙJ	Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang	Valid

Tabel 6.10 Validitas Enkripsi dan Dekripsi (Lanjutan)

Ukuran *File* dan Isi *File*	*Plaintext*	Enkripsi	Dekripsi	Status
	tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Data yang sudah dibuat di Microsoft Office kemudian disimpan	T¦xûzgÎµ2·ú_¬ó¤îçl¾ý ñP_ Ó$¯¤²Ä¶_ ÑìüØÒ_¬UÐ _Ø#,__pq(ÖÚ¬Ú('Ç'ð? ÈW__}w_C_ ØdkÒä_Õëåyïòý÷ØºÊ ©'º¿¨ê¯sLðÂñÒ¦_£J.ë ¡Ò¯Åå_Ô_+[~IF_ Æ¨ÃÊQÔïc´SÛn_"ÂQ\(lµ\Wu5'¦Ñm3!¹,[= ëTàëN^*¶øi_`/t-_¢¬Ú ÕäûÇ#µðX)&fÑ}ÈÐq¯Ý ,S_o½õ__`_£__ =«4M¦JÚXÊ[¦nÛ-ðô_² Ñlº} Ô_½ÚY\p7_áb_Ñ/__l¸ Ó üÍ¨ÿÐB_§ûá_°R¸LÍ- ¢û ã_ºX¦<ß{ê,:ÀF:7_{ V_høµ_±¶ù_Ý:[áé(_ e¢ÙtúA¨)+{´²´À4{_µÌÌë #öË^ŞéÇÞ_ñO{ÐOÈd_ º%ó¾___!2ÄÌÒÚ/¹__5 û¥cÍ÷¾¢Ã¬c£pçµa&¯[Q_Ko-Ûu¾_öZ-O¤_ ·»K*x,¹•_<§è¶þÌj_	mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Data yang sudah dibuat di Microsoft Office	

Tabel 6.10 Validitas Enkripsi dan Dekripsi (Lanjutan)

Ukuran *File* dan Isi *File*	*Plaintext*	Enkripsi	Dekripsi	Status
	dalam ekstensi aslinya dalam PDF. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF.		kemudian disimpan dalam ekstensi aslinya dalam PDF. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF.	
4 KB, 13 baris	Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat	¤>~_!bO¯¸G¶_k_V_-T ÁàµSURtz`õ6÷MIý¦{_ _%øt/._Ù+cõ{¼Äb-_±' Åë¶k+_*3Vþ_`Î¨J;Q{+ ôÄû#÷>- Ð_G_•3/Úh£ÛÐÐH_ Ò@ò_÷_ëf_glë_§vï_¥ PÛÜ¼÷àÎ«Äò<&D·[ý_P U_£_¼_OýïõýÛ@S¨_U ùq$©þ_6üèCHÐJD È2~AF~YÄ_ fn§Í3e}ÌîÂG}mgæE_âT 8v;¯ÑCc¯ÊF³IÉJ¹jÿGPÓ _*¶»ÙJ T¦xûzgÎµ2·ú_¬ó¤îçI¾ý ñP_ Ó$¯¤²Ä¶_ ÑìüØÒ_¬UÐ _Ø#,__pq(ÖÚ¬Ú('Ç'ð? ÈW__}w_C_ ØdkÒä_Õëåyïòý÷ØºÊ ©'º¿¨ê¯sLðÂñÒ¦_£J.ë ¡Ò¯Åå_Ô_+[~IF_ Æ¨ÃÊQÔïc´SÛn_"ÂQ\(lµ\Wu5'|Ñm3!¹,[= ëTàëN^*¶øi_`/t-_¢¬Ú	Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam	Valid

Tabel 6.10 Validitas Enkripsi dan Dekripsi (Lanjutan)

Ukuran *File* dan Isi *File*	*Plaintext*	Enkripsi	Dekripsi	Status
	ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat	ÕäûÇ#µðX)&fÑ}ÈÐq¯Ý ,S_o½õ__`_£__ =«4M|JÚXÊ[|nÛ-ðô_²Ñlº} Ô_½ÚY^- _½Ókk____tÔ¹0K_,?.ï |2Í_MA V^ØýÐ6_*Ø_r Ý~ö¨ÆTarï&¤¯x¾Ñ_ÑÎ Cöá_¶_·£[Øy_i__g{Û_ _¹uêî`£n Ñ¯øïË__j³_ª£¡ÀáÖjÝI·ë _8y:@Ý£f#ã]_mÚ$#`q ³¥×Æ¤°Ç RÂ___A§>÷^¯ÃDMæ7 _8+KmqÁ__)J_kÒ'Ñm *- ©ø_3T5ý8L±ësXto__ã .-K!4<_=Cèp¡2òÂ æèOþ¥1¡Ú¥GåÜ4÷W, +_¶ä¦×üÖà`vfZep'c_j 5o3QR½Û_ ¨ÚëÐô_?O_W_}h¿x% _c®þ anà__V_,_{(ñ©Y©_ ¤¤&'eØpSÔ`}BêÃìEÍ¢& _R ¸-¤ÆÍo÷!®½'M(_nÚ£¡ Ú?´$-è=jCæ!QÏaØ__µ R=A_û^kuN)`z_¹þ{8½` _{ãúËD³or&¯0}t´ÌÐQð _W4ÑHan+uÓS6ø_ ËrZ^òË/ðÑ_Þ¿ - ÆY²»ñòÏuç%~>ªÀÕ•E Az»_Èåf¢I±óLdözÆ¯Ïr &_«_Ø«_?E6õB¨4ÿ3Ð à@_ÔÎÛác5~·Ý3ÖQ?- ä_Z_ F-~§ç4`_÷P_$[_AØ_¶ë fZ-¹0+ê_f!áñlN¶°½ÉwS	PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut.	

Tabel 6.10 Validitas Enkripsi dan Dekripsi (Lanjutan)

Ukuran *File* dan Isi *File*	*Plaintext*	Enkripsi	Dekripsi	Status
	di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya	y_KĐ¯vÖ[?PmFà_!º î_¿AÁ¾/«h	Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya. Dalam pengolahan data terdapat banyak aplikasi yang mendukung hal tersebut. Data yang sudah dibuat di Microsoft Office kemudian disimpan dalam ekstensi aslinya dalam PDF. Pada kemajuan teknologi saat ini, orang semakin dimudahkan untuk menyelesaikan pekerjaannya.	

Dari pengujian diatas dapat kesimpulan yaitu pada Tabel 6.10 menunjukkan perubahan pada saat *file* tersebut masuk pada proses enkripsi, sehingga menampilkan kararakter acak (*chipertext*). Setelah itu ketika *file* sudah dalam bentuk *chipertext*, *file* akan masuk pada proses dekripsi sehingga menjadi *file plaintext* awal. Hal ini menunjukkan bahwa validitas enkripsi dan dekripsi pada ketiga *file* dengan ukuran file 4 KB dengan 3 baris, 8 baris dan 13 baris menggunakan algoritme SPECK *bock cipher* adalah valid.

6.5 Pengujian *Share* dan *Reconstruct File*

Pengujian *share* dan *reconstruct* dilakukan untuk mengetahui apakah *file* tersebut benar-benar dapat melewati proses *share* dan *reconstruct* dengan baik. Sehingga dapat diketahui bahwa algoritme Shamir's *secret Sharing* yang digunakan dapat berjalan dengan baik. Pengujian ini dilakukan dengan cara menjalankan proses *share* dan *reconstruct* pada nilai *cipher* yang sudah dihasilkan algoritme SPECK *block cipher*. Ukuran *file* yang digunakan adalah 4 KB dengan isi 3 baris dan isi tersebut dirubah menjadi bilangan desimal. Jumlah *share* dan *reconstruct* yang digunakan adalah 5 untuk *share* dan 4 untuk maksimal *reconstruct*. Pengujian *share file* dapat dilihat pada Tabel 6.11, sedangkan *reconstruct file* dapat dilihat pada Tabel 6.12.

Tabel 6.11 *Share file*

***Ciphertext* ASCII**
åP±_î_2Ã>ÄÔ¦e_±__¿RJ!V¸ÅÌ·_ySýe%`q;wêU)9"_jb3u'¤È±SõZ¢Ò6« Ï_]- #CfàJhïØèà_¾_»¥ú]L_~T):ùöEÚ%¡Íý_{_(_x"ñ_ *¸1÷_¾_ÔÈw³:¸_²2³üKP3x²bÈ¶êM½U_k¹ þâï²Ô¨v_Ìh_A>èÃ/ _ð!{÷_®Ð¡3=Ü=ã{Ëî øø¢6÷Ø¤|dì`ñ«_Ð8_d_ì)Î°ã1_Éú_¤Âó·<_'_æðûØó®È m/×HsäÖf{²|KTg¤PÒ«© a
***Ciphertext* Biner**
11100101100000010101000010110001000000101110111000010111 00110010110000110011111011000100110101001010011001100101 00000001100010101011000100001110000101011011111101010010 01001010001000010101011010111000110001011100110010110111 00011001011110011001111100101001111111101100100110110010 1 00100101011000000111000100111011011101111110101001010101 10000011100111100010100100111001001000100001111101101010 01100010001100110111010100100111101001001100100010110001 01010011111101010101101010100010110100100011011010101011 10100000100001101100111110011100000010110101110100101101 10011011001000110100001101100110100100101110000001001010 01101000111011111101100011101000111000000001101010111110 00011001101110111010010111111010010111010100110000011011 01111110100111100101010000101001001110101000000011000100 01111100111110110010001011000011111011010001001010110100 11100111111111101000010110111101100011100001010001001101 00000110001111000001000101111000100011000001010101011100 01001111000110001111101110001001010111110000011001101010 01100100001110111101100110011101010111000000010001011001

Tabel 6.11 *Share file* (lanjutan)

***Ciphertext* Biner**
000110010101100111000100110000100111111001000011001001011010100000011001101111000100100001011001001100001011001000101101101110101001001101101111010101010100000001011010111001111000010111011111011001011010010101000011101100001111111001100011010000000100100000100000011000101100111110100001011001100111101000010110011001111011101110000111101111000111011110111101000011111000111111010100000111110001111100010100010001101101111011111010110001010010001111100011001001110110001100000111100110010011111111010000011101010011001100111010001101011110111000110010110010011111000100101101010100011001111010010001010000110100101010101110101001100000110000110
Share File
Share 1
538314688161368442174058936439581536212381916822050340361212923525486398086620907083903883588866692984595705275454292966030688901490530722284067012514263776389941251129114435254808429532918549014210101135270117663523776585885023847833816473242810020731472855612001452615196861705381128438056272664793893855842945012332995366088152401573241889067963015050620713361569571368440714556933194655666156993321251328720603125239146511239365702603341797888189636220613969585127813500842901342130037681618340396220150764280157252969212767054617681103102461228548075624872593139375468439022108264601171539540550666388246490326086600035528550563069589789702609 9
Share 2
338762921166841453534310832880710594730256272246096078256916917673271792293055351771941452715404417991599021162815332325422187716352837285177323303141395184367360022748449028949343336687978075231048627444752434554783227882549659424069357077007919548924399396701734633360727678321965768964614063051062715586512658954380899328335314627738196414797196831307095037458998270339044881266206612432362400579277713108398978437956320987548155218403961994345149131640370760283916007130888514674973259152471416439796372746680639856193493143385201862550357717092053952972

Tabel 6.11 *Share file* (lanjutan)

Share 2
924318980993530849089286197920816758108928803314971729361150134846548020631715784408371434 5
Share 3
157542846896610196579329190503337701005395027183287183516246248450223712019161922519582471685516441239022136305974309062552693853338150176180194028248702203965092310206778540138932670670513529419454888801791748713335916500795204975264208460819401791940696069695712224476093891018549838021363438424405454438447528305740522675513163697005995956575101262942848586372385082739378722840345181597381225315658966916426796438365438886775290437689641934364378191378083243460463118271600557351007521400887621551288127102135570774003589898258071505947517925481759733572684635092873460422858781998424967890406910893234309740095617733081380726628842784506976098 2
Share 4
442474299247314805896234129740090142114413107747178227737041873770946864503555183583133536042034189932161586941244348421944192668200456739073450318875833611942511081826113133833467577825573055636293415111274065604595367797459840551499749064584511320133622610785445405221624707635134478547921228810674276169117242247788426637760325923170950482304335079199322910469813781709982889549618599374077468901615428696105171751082613363084079953490262130821337686797840034159251311901646170683850742871740697594864349084536053377227870274588655687394773181345265610920736360934491522832925959931744613108975289030161034979130681267892400196697488979125644922 8
Share 5
261254224977083548941252487362717248389551862684369332996371204547898784229661754330774555012146213179584702084403325159074698805185769630076321043983140631540243369284442645023056911808108509824699676468313379763148056415705386102694600448395993563149919283779422996336990920331718547604670604184017015021052111599148049984938174992438750024082239510835076459383200594110316731123757168539096293637996682504132989751491731262311215172775942070840566746535552517335798423042358213359885005120156902706356103439990984295037967029461525330791933389734971391520496677046371452406695455732248764241273271120080372989865148866122723290269461597922424958658
Keterangan : Valid

Dari hasil pengujian *share file* pada Tabel 6.11, dapat disimpulkan bahwa sistem dapat memecah *file* menjadi beberapa bagian dan dengan hasil valid. Dari

hasil pengujian *reconstruct* pada Tabel 6.12 dapat disimpulkan bahwa dengan menggunakan kombinasi 2 bagian (*share*) pada *file*, sistem mampu mengembalikan *ciphertext* yang sama. Sehingga pada pengujian *reconstruct* ini dapat dikatakan valid.

Tabel 6.12 *Reconstruct file*

No	Kombinasi	Total Reconstruct	Status	Keterangan
1	*share* 1, *share* 2	2	Terpenuhi	Valid
2	*share* 1, *share* 3	2	Terpenuhi	Valid
3	*share* 1, *share* 4	2	Terpenuhi	Valid
4	*share* 1, *share* 5	2	Terpenuhi	Valid
5	*share* 2, *share* 3	2	Terpenuhi	Valid
6	*share* 2, *share* 4	2	Terpenuhi	Valid
7	*share* 2, *share* 5	2	Terpenuhi	Valid
8	*share* 3, *share* 4	2	Terpenuhi	Valid
9	*share* 3, *share* 5	2	Terpenuhi	Valid
10	*share* 4, *share* 5	2	Terpenuhi	Valid
11	*share* 1, *share* 2, *share 3*	3	Terpenuhi	Valid
12	*share* 1, *share* 2, *share* 4	3	Terpenuhi	Valid
13	*share* 1, *share* 2, *share* 5	3	Terpenuhi	Valid
14	*share* 1, *share* 3, *share* 4	3	Terpenuhi	Valid
15	*share* 1, *share* 3, *share* 5	3	Terpenuhi	Valid
16	*share* 1, *share* 4, *share* 5	3	Terpenuhi	Valid
17	*share* 2, *share* 3, *share* 4	3	Terpenuhi	Valid
18	*share* 2, *share* 3, *share* 5	3	Terpenuhi	Valid
19	*share* 2, *share* 4, *share* 5	3	Terpenuhi	Valid
20	*share* 3, *share* 4, *share* 5	3	Terpenuhi	Valid
21	*share* 1, *share* 2, *share* 3, *share* 4	4	Terpenuhi	Valid
22	*share* 1, *share* 2, *share* 3, *share* 5	4	Terpenuhi	Valid
23	*share* 1, *share* 2, *share* 4, *share* 5	4	Terpenuhi	Valid
24	*share* 1, *share* 3, *share* 4, *share* 5	4	Terpenuhi	Valid
25	*share* 2, *share* 3, *share* 4, *share* 5	4	Terpenuhi	Valid

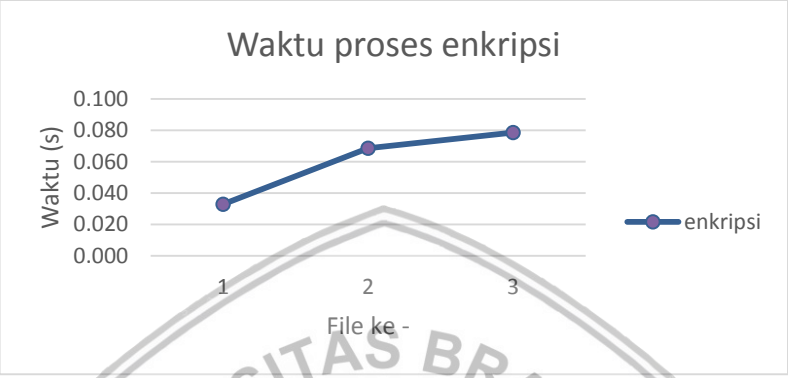
6.6 Pengujian Waktu

6.6.1 Pengujian Waktu Proses Enkripsi

Pengujian waktu enkripsi menggunakan parameter kunci yang sama yaitu "abcd1234" melakukan pengujian pada *file* yang masing-masing berukuran 4 KB dengan jumlah baris yang berbeda. Pengujian menggunakan implementasi SPECK 128/128 dilakukan sebanyak 1967 kali. Skenario pengujian ini terbagi menjadi 3 yang rata-rata hasilnya dapat dilihat pada Tabel 6.13.

Tabel 6.13 Hasil Pengujian Waktu Proses Enkripsi

File ke -	Jumlah baris	Hasil (s)
1	3	0,033
2	8	0,069
3	13	0,074



Gambar 6.3 Pengujian Waktu Enkripsi

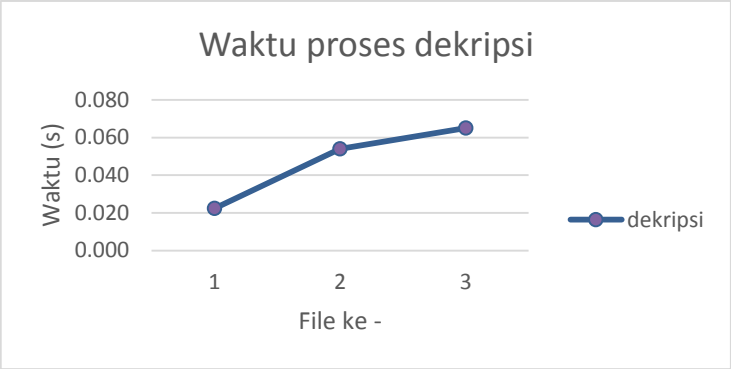
Pada Tabel 6.13 dan Gambar 6.3 dapat dilihat rata-rata waktu enkripsi terlama adalah pada *file* ke-3 dengan waktu 0,074*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa proses enkripsi membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu.

6.6.2 Pengujian Waktu Proses Dekripsi

Pengujian waktu dekripsi menggunakan parameter kunci yang sama yaitu "abcd1234" melakukan pengujian pada *file* yang masing-masing berukuran 4 KB dengan jumlah baris yang berbeda. Pengujian menggunakan implementasi SPECK 128/128 dilakukan sebanyak 1967 kali. Skenario pengujian ini terbagi menjadi 3 yang rata-rata hasilnya dapat dilihat pada Tabel 6.14.

Tabel 6.14 Hasil Pengujian Waktu Proses Dekripsi

File ke -	Jumlah baris	Hasil (s)
1	3	0,022
2	8	0,054
3	13	0,065



Gambar 6.4 Pengujian Waktu Dekripsi

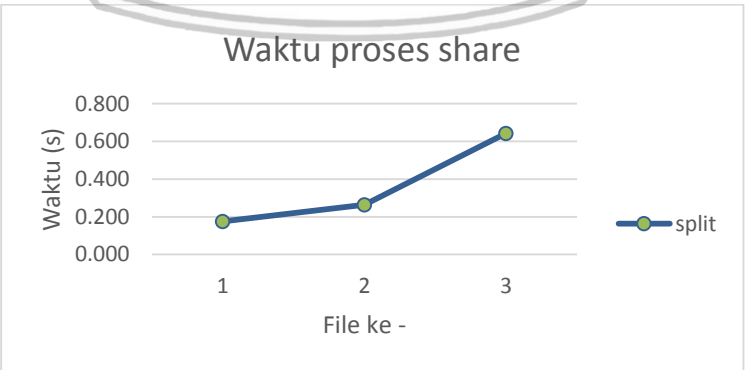
Pada Tabel 6.14 dan Gambar 6.4 dapat dilihat rata-rata waktu dekripsi terlama adalah pada *file* ke-3 dengan waktu 0,065*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa proses dekripsi membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu.

6.6.3 Pengujian Waktu Proses *Share*

Pengujian waktu pada proses *share* pada implementasi Shamir's *secret sharing* dengan banyak pengujian sebanyak satu kali, dengan banyak kombinasi yaitu 1967 kombinasi kombinasi angka *reconstruct*. *File* yang diuji masing-masing berukuran 4 KB dengan jumlah baris yang berbeda. Skenario pengujian ini terbagi menjadi 3 yang hasilnya dapat dilihat pada Tabel 6.15.

Tabel 6.15 Hasil Pengujian Waktu Proses *Share*

File ke -	Jumlah baris	Hasil (s)
1	3	0,175
2	8	0,263
3	13	0,647



Gambar 6.5 Pengujian Waktu *Share*

Pada Tabel 6.15 dan Gambar 6.5 dapat dilihat rata-rata waktu *share* terlama adalah pada *file* ke-3 dengan waktu 0,647*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa proses *share* membutuhkan lebih banyak waktu ketika

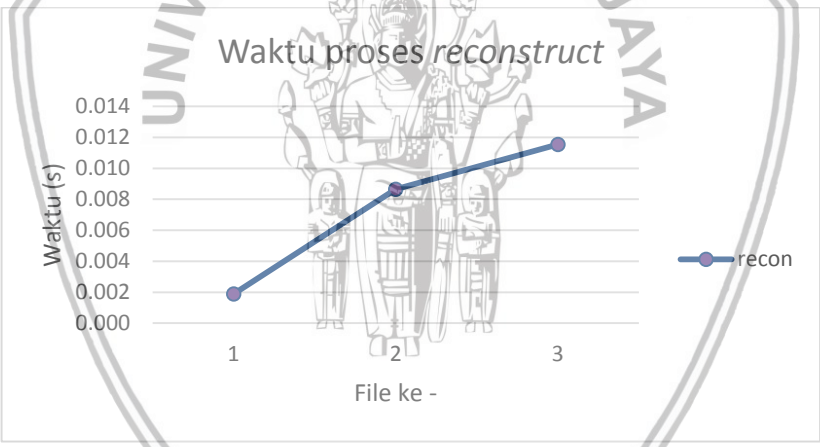
jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu. Sebelum masuk ke proses *share* diperlukan untuk menghitung bilangan prima terlebih dahulu, jadi semakin banyak kata yang ada semakin lama waktu yang dibutuhkan untuk proses *share*.

6.6.4 Pengujian Waktu Proses *Reconstruct*

Pengujian waktu pada proses *reconstruct* pada implementasi Shamir's *secret sharing* dengan banyak pengujian sebanyak satu kali, dengan banyak 1967 kombinasi angka *reconstruct*. *File* yang diuji masing-masing berukuran 4 KB dengan jumlah baris yang berbeda. Skenario pengujian ini terbagi menjadi 3 yang hasilnya dapat dilihat pada Tabel 6.16.

Tabel 6.16 Hasil Pengujian Waktu Proses *Reconstruct*

File ke -	Jumlah baris	Hasil (s)
1	3	0,002
2	8	0,009
3	13	0,012



Gambar 6.6 Pengujian Waktu *Reconstruct*

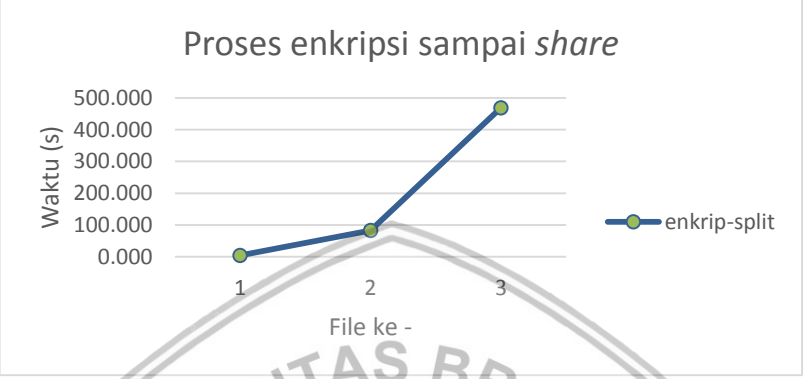
Pada Tabel 6.16 dan Gambar 6.6 dapat dilihat rata-rata waktu *reconstruct* terlama adalah pada *file* ke-3 dengan waktu 0,0125*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa proses *reconstruct* membutuhkan lebih banyak waktu ketika jumlah kombinasi *share* semakin banyak.

6.6.5 Pengujian Waktu Proses Enkripsi Sampai *Share*

Pengujian waktu pada proses enkripsi sampai *share file* menggunakan parameter kunci yang sama yaitu "abcd1234" melakukan pengujian pada *file* yang masing-masing berukuran 4 KB dengan jumlah baris yang berbeda. Pengujian menggunakan implementasi SPECK 128/128 dan Shamir's *secret sharing* yang dilakukan sebanyak 1967 kali. Skenario pengujian ini terbagi menjadi 3 yang rata-rata hasilnya dapat dilihat pada Tabel 6.17.

Tabel 6.17 Hasil Pengujian Waktu Proses Enkripsi Sampai *Share*

File ke -	Jumlah baris	Hasil (s)
1	3	4,200
2	8	82,911
3	13	469,056



Gambar 6.7 Pengujian waktu proses enkripsi sampai *share*

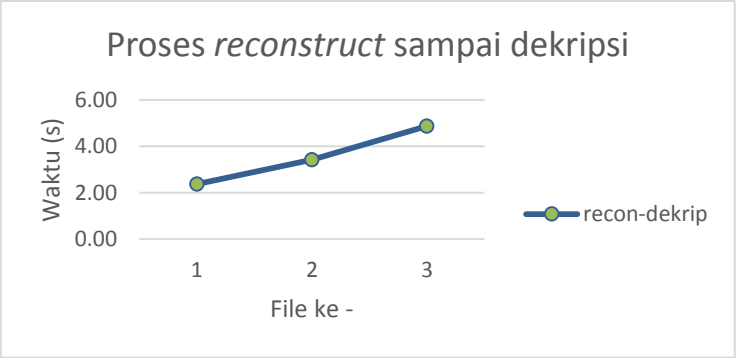
Pada Tabel 6.17 dan Gambar 6.7 dapat dilihat rata-rata waktu enkripsi sampai *share* terlama adalah pada *file* ke-3 dengan waktu 469,056*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa proses enkripsi sampai *share* membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu. Sebelum masuk ke proses *share* diperlukan untuk menghitung bilangan prima terlebih dahulu, jadi semakin banyak kata yang ada semakin lama waktu yang dibutuhkan untuk proses *share*.

6.6.6 Pengujian Waktu Proses *Reconstruct* Sampai Dekripsi

Pengujian waktu pada proses *reconstruct* sampai dekripsi *file* pada implementasi algoritme SPECK *block* cipher dan Shamir's *secret sharing* pada *file* teks. Pengujian dilakukan sebanyak 1967 kali. *File* yang diuji berukuran 4 KB dengan perbedaan baris yaitu 3, 8 dan 13. Skenario pengujian ini terbagi menjadi 3 yang hasilnya dapat dilihat pada Tabel 6.18.

Tabel 6.18 Hasil Pengujian Waktu Proses *Reconstruct* Sampai Dekripsi

File ke -	Jumlah baris	Hasil (s)
1	3	2,365
2	8	3,413
3	13	4,861



Gambar 6.8 Pengujian waktu proses *reconstruct* sampai dekripsi

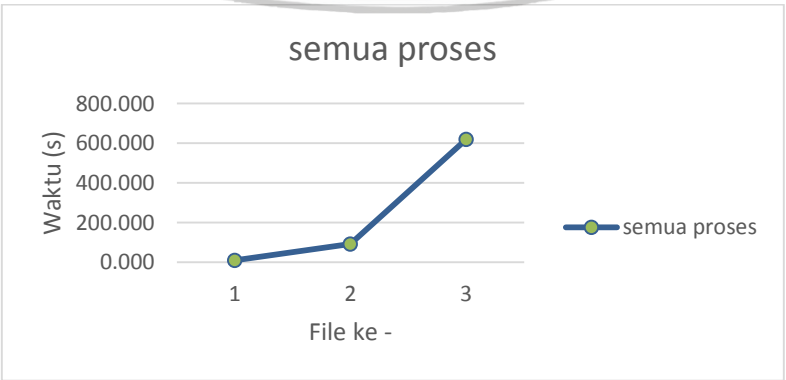
Pada Tabel 6.18 dan Gambar 6.8 dapat dilihat rata-rata waktu proses *reconstruct* sampai dekripsi terlama adalah pada *file* ke-3 dengan waktu 4,861*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa proses proses *reconstruct* sampai dekripsi membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu.

6.6.7 Pengujian Waktu Semua proses

Pengujian waktu semua proses pada implementasi algoritme SPECK *block* cipher dan Shamir's *secret sharing* pada *file* teks. Pengujian dilakukan sebanyak 1967 kali. *File* yang diuji berukuran 4 KB dengan perbedaan baris yaitu 3, 8 dan 13. Skenario pengujian ini terbagi menjadi 3 yang hasilnya dapat dilihat pada Tabel 6.19.

Tabel 6.19 Hasil Pengujian Waktu Semua Proses

File ke -	Jumlah baris	Hasil (s)
1	3	9,433
2	8	91,938
3	13	619,036



Gambar 6.9 Pengujian waktu semua proses

Pada Tabel 6.19 dan Gambar 6.9 dapat dilihat rata-rata waktu keseluruhan proses terlama adalah pada *file* ke-3 dengan waktu 619,036 *s*. Berdasarkan hasil

tersebut dapat disimpulkan bahwa saat menjalankan semua proses membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu.

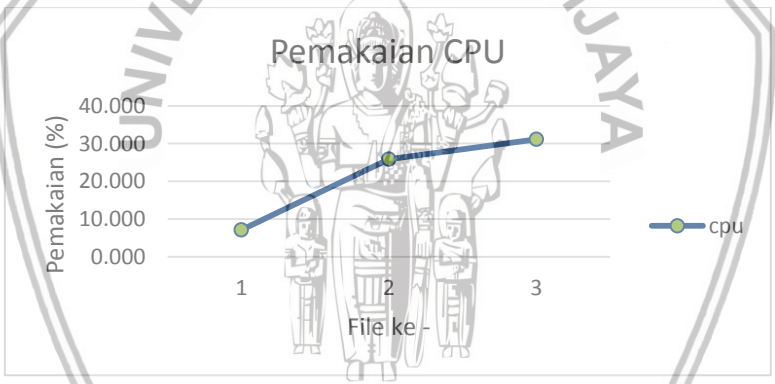
6.7 Pengujian Pemakaian CPU dan RAM

6.7.1 Pengujian Pemakaian CPU

Berikut merupakan pengujian banyaknya pemakaian CPU pada implementasi algoritme SPECK *block* cipher dan Shamir's *secret sharing* pada *file* teks. Pengujian dilakukan sebanyak 1967 kali. *File* yang diuji berukuran 4 KB dengan perbedaan baris yaitu 3, 8 dan 13. Skenario pengujian ini terbagi menjadi 3 yang hasilnya dapat dilihat pada Tabel 6.20.

Tabel 6.20 Hasil Pengujian Pemakaian CPU

File ke -	Jumlah baris	Hasil (%)
1	3	7,058
2	8	25,885
3	13	31,095



Gambar 6.10 Pengujian pemakaian CPU

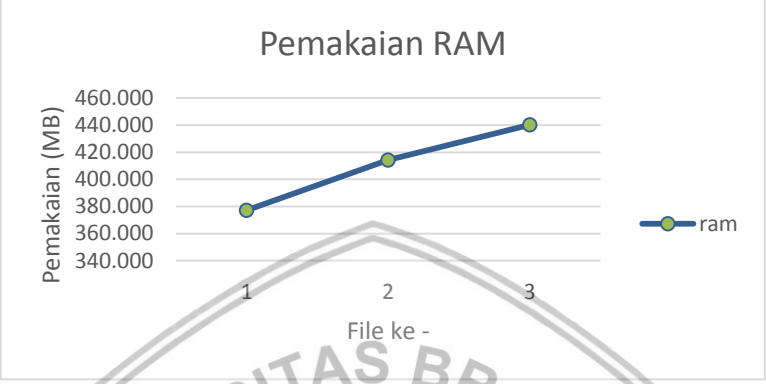
Pada Tabel 6.20 dan Gambar 6.10 dapat dilihat rata-rata pemakaian CPU pada *file* ke-3 dengan waktu 31,095*s*. Berdasarkan hasil tersebut dapat disimpulkan bahwa CPU membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu sehingga semakin banyak juga pemakaian CPU yang diperlukan untuk menjalankan sistem.

6.7.2 Pengujian pemakaian RAM

Berikut merupakan pengujian banyaknya pemakaian RAM pada implementasi algoritme SPECK *block* cipher dan Shamir's *secret sharing* pada *file* teks. Pengujian dilakukan sebanyak 1967 kali. *File* yang diuji berukuran 4 KB dengan perbedaan baris yaitu 3, 8 dan 13. Skenario pengujian ini terbagi menjadi 3 yang hasilnya dapat dilihat pada Tabel 6.21.

Tabel 6.21 Hasil Pengujian Pemakaian RAM

File ke -	Jumlah baris	Hasil (MB)
1	3	377,277
2	8	414,283
3	13	440,231



Gambar 6.11 Pengujian pemakaian RAM

Pada Tabel 6.21 dan Gambar 6.11 dapat dilihat rata-rata pemakaian RAM pada *file* ke-3 dengan banyak 440,231 MB. Berdasarkan hasil tersebut dapat disimpulkan bahwa RAM membutuhkan lebih banyak waktu ketika jumlah baris semakin banyak, hal ini dikarenakan setiap kata pada *file* diproses satu persatu sehingga semakin banyak juga pemakaian RAM yang diperlukan untuk menjalankan sistem.

6.8 Analisis Pengujian Waktu Eksekusi

Data hasil pengujian akan dianalisis untuk menilai kinerja Algoritme SPECK dan Shamir's. Pertama melakukan pengujian Kolmogorov Smirnov, pengujian ini digunakan untuk menentukan nilai signifikansi data tersebut berdistribusi normal atau tidak berdistribusi normal. Nilai Signifikansi dikatakan normal jika > 0,05 dan berdistribusi tidak normal jika nilainya <0,05. Pengujian Anova dilakukan jika nilai signifikansi data berdistribusi normal, sedangkan pengujian Kruskal-Wallis dilakukan jika nilai signifikansi data tidak berdistribusi normal. Apabila nilai signifikansi yang dihasilkan dalam pengujian Kruskal-Wallis < 0,05 maka akan dilakukan pengujian Post Hoc *Test*. Pengujian Post Hoc *Test* bertujuan untuk mencari data kelompok manakah yang menyebabkan terjadinya perbedaan waktu yang tinggi dengan melihat nilai signifikansi dan *Chi-Square* tertinggi yang dihasilkan. Data kelompok yang akan dibandingkan adalah data 1 dengan data 2, data 1 dengan data 3 dan data 2 dengan data 3. Nilai data 1 yaitu *share* 5 *reconstruct* 2, data 2 yaitu *share* 5 *reconstruct* 3 dan data 3 yaitu *share* 5 *reconstruct* 4.

6.8.1 Pengujian Kolmogorov-Smirnov

Pengujian dilakukan dengan menganalisis nilai residual dari keseluruhan data *share* 5 *reconstruct* 2, *share* 5 *reconstruct* 3 dan *share* 5 *reconstruct* 4 dengan jumlah 15 data untuk masing-masing waktu enkripsi, dekripsi, *share*, *reconstruct*, proses enkripsi sampai *share*, proses *reconstruct* sampai dekripsi, semua proses, serta banyaknya pemakaian CPU dan RAM. Setelah itu akan diambil keputusan berdasarkan nilai signifikansi yang dihasilkan oleh proses tersebut.

6.8.1.1 Pengujian File 4KB (3 baris)

Tabel 6.22 Pengujian Kolmogorov-Smirnov pada enkripsi *file* (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		enkripsi
N		15
Normal Parameters[a,b]	Mean	.0333
	Std. Deviation	.00519
Most Extreme Differences	Absolute	.463
	Positive	.463
	Negative	-.331
Test Statistic		.463
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.22 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu enkripsi pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,000.

Tabel 6.23 Pengujian Kolmogorov-Smirnov pada dekripsi *file* (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		dekripsi
N		15
Normal Parameters[a,b]	Mean	.0207
	Std. Deviation	.00965
Most Extreme Differences	Absolute	.421
	Positive	.421
	Negative	-.276
Test Statistic		.421
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.23 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu dekripsi pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,000.

Tabel 6.24 Pengujian Kolmogorov-Smirnov pada *share file* (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		share
N		15
Normal Parameters[a,b]	Mean	.1353
	Std. Deviation	.04299
Most Extreme Differences	Absolute	.130
	Positive	.130
	Negative	-.085
Test Statistic		.130
Asymp. Sig. (2-tailed)		.200[c,d]

Pada Tabel 6.24 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu *share file* pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,200.

Tabel 6.25 Pengujian Kolmogorov-Smirnov pada *reconstruct* (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		reconstruct
N		15
Normal Parameters[a,b]	Mean	.0000
	Std. Deviation	.00000[c]

Pada Tabel 6.25 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu *reconstruct file* pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,000.

Tabel 6.26 Pengujian Kolmogorov-Smirnov pada proses enkripsi sampai *share* (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		enkripshare
N		15
Normal Parameters[a,b]	Mean	3.8583
	Std. Deviation	1.23777
Most Extreme Differences	Absolute	.300
	Positive	.300
	Negative	-.166
Test Statistic		.300
Asymp. Sig. (2-tailed)		.001[c]

Pada Tabel 6.26 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu proses enkripsi sampai *share* pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,001.

Tabel 6.27 Pengujian Kolmogorov-Smirnov pada proses *reconstruct* sampai dekripsi (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		recondekrip
N		15
Normal Parameters[a,b]	Mean	2.1179
	Std. Deviation	.94073
Most Extreme Differences	Absolute	.133
	Positive	.133
	Negative	-.096
Test Statistic		.133
Asymp. Sig. (2-tailed)		.200[c]

Pada Tabel 6.27 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu proses *reconstruct* sampai dekripsi pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,200.

Tabel 6.28 Pengujian Kolmogorov-Smirnov pada semua proses (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		allproses
N		15
Normal Parameters[a,b]	Mean	8.8053
	Std. Deviation	1.51078
Most Extreme Differences	Absolute	.181
	Positive	.181
	Negative	-.157
Test Statistic		.181
Asymp. Sig. (2-tailed)		.200[c,d]

Pada Tabel 6.28 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu pada proses semua proses pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,200.

Tabel 6.29 Pengujian Kolmogorov-Smirnov pada CPU (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		cpu
N		15
Normal Parameters[a,b]	Mean	5.9400
	Std. Deviation	1.22812
Most Extreme Differences	Absolute	.182
	Positive	.182
	Negative	-.091
Test Statistic		.182
Asymp. Sig. (2-tailed)		.197[c]

Pada Tabel 6.29 menunjukkan hasil pengujian Kolmogorov-Smirnov Test CPU pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,197.

Tabel 6.30 Pengujian Kolmogorov-Smirnov pada RAM (3 baris)

One-Sample Kolmogorov-Smirnov Test		
		ram
N		15
Normal Parameters[a,b]	Mean	476.7067
	Std. Deviation	4.76632
Most Extreme Differences	Absolute	.259
	Positive	.259
	Negative	-.180
Test Statistic		.259
Asymp. Sig. (2-tailed)		.008[c]

Pada Tabel 6.30 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu proses enkripsi sampai *share* pada *file* 4 KB (3 baris) dengan nilai signifikansi 0,000.

Pengambilan keputusan:

H0: nilai signifikansi > 0,05 maka nilai residual berdistribusi normal

H1: nilai signifikansi < 0,05 maka nilai residual tidak berdistribusi normal

Dari hasil pengujian Kolmogorov Smirnov diatas, nilai signifikansi yang dihasilkan oleh nilai residual waktu enkripsi, dekripsi, *reconstruct*, proses enkripsi *share*, dan RAM < 0,05 yang berarti keputusan H0 ditolak karena nilai residual tidak berdistribusi normal. Sedangkan nilai signifikansi yang dihasilkan oleh nilai residual *share*, proses *reconstruct* dekripsi, semua proses, dan CPU > 0.05 yang berarti keputusan H0 diterima karena nilai residual berdistribusi normal. Apabila nilai residual berdistribusi normal, maka langkah selanjutnya dalah melakukan pengujian Anova untuk membandingkan nilai rata-rata dari semua kelompok data. Apabila nilai residual berdistribusi tidak normal, maka langkah selanjutnya adalah melakukan pengujian Kruskal-Wallis untuk mencari perbedaan waktu antar data.

6.8.1.2 Pengujian File 4KB (8 baris)

Tabel 6.31 Pengujian Kolmogorov-Smirnov enkripsi (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Enkripsi
N		15
Normal Parameters[a,b]	Mean	.0661
	Std. Deviation	.00877
Most Extreme Differences	Absolute	.503
	Positive	.503
	Negative	-.321
Test Statistic		.503
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.31 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu enkripsi *file* pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,000.

Tabel 6.32 Pengujian Kolmogorov-Smirnov dekripsi (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Dekripsi
N		15
Normal Parameters[a,b]	Mean	.0471
	Std. Deviation	.01451
Most Extreme Differences	Absolute	.232
	Positive	.168
	Negative	-.232
Test Statistic		.232
Asymp. Sig. (2-tailed)		.030[c]

Pada Tabel 6.32 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu dekripsi *file* pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,030.

Tabel 6.33 Pengujian Kolmogorov-Smirnov *share* (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Share
N		15
Normal Parameters[a,b]	Mean	.2531
	Std. Deviation	.22762
Most Extreme Differences	Absolute	.249
	Positive	.249
	Negative	-.202
Test Statistic		.249
Asymp. Sig. (2-tailed)		.013[c]

Pada Tabel 6.33 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu *share file* pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,013.

Tabel 6.34 Pengujian Kolmogorov-Smirnov *reconstruct* (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Reconstruct
N		15
Normal Parameters[a,b]	Mean	.0052
	Std. Deviation	.00762
Most Extreme Differences	Absolute	.419
	Positive	.419
	Negative	-.247
Test Statistic		.419
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.34 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu *reconstruct file* pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,000.

Tabel 6.35 Pengujian Kolmogorov-Smirnov proses enkripsi sampai *share* (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Enkripshare
N		25
Normal Parameters[a,b]	Mean	64.8202
	Std. Deviation	48.87358
Most Extreme Differences	Absolute	.153
	Positive	.153
	Negative	-.136
Test Statistic		.153
Asymp. Sig. (2-tailed)		.136[c]

Pada Tabel 6.35 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu proses enkripsi sampai *share file* pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,136.

Tabel 6.36 Pengujian Kolmogorov-Smirnov proses *reconstruct* sampai dekripsi (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Recondekrip
N		15
Normal Parameters[a,b]	Mean	2.5429
	Std. Deviation	.75183
Most Extreme Differences	Absolute	.148
	Positive	.148
	Negative	-.109
Test Statistic		.148
Asymp. Sig. (2-tailed)		.200[c,d]

Pada Tabel 6.36 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu proses *reconstruct* sampai dekripsi *file* pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,200.

Tabel 6.37 Pengujian Kolmogorov-Smirnov semua proses (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Allproses
N		15
Normal Parameters[a,b]	Mean	71.5115
	Std. Deviation	49.16768
Most Extreme Differences	Absolute	.177
	Positive	.177
	Negative	-.156
Test Statistic		.177
Asymp. Sig. (2-tailed)		.200[c,d]

Pada Tabel 6.37 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu semua proses pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,200.

Tabel 6.38 Pengujian Kolmogorov-Smirnov CPU (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Cpu
N		15
Normal Parameters[a,b]	Mean	19.0467
	Std. Deviation	12.37277
Most Extreme Differences	Absolute	.291
	Positive	.263

Tabel 6.38 Pengujian Kolmogorov-Smirnov CPU (8 baris) (lanjutan)

One-Sample Kolmogorov-Smirnov Test		
		Cpu
	Negative	-.291
Test Statistic		.291
Asymp. Sig. (2-tailed)		.001[c]

Pada Tabel 6.38 menunjukkan hasil pengujian Kolmogorov-Smirnov Test pemakaian CPU pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,001.

Tabel 6.39 Pengujian Kolmogorov-Smirnov RAM pada file 4 KB (8 baris)

One-Sample Kolmogorov-Smirnov Test		
		Ram
N		15
Normal Parameters[a,b]	Mean	410.6800
	Std. Deviation	12.29775
Most Extreme Differences	Absolute	.314
	Positive	.162
	Negative	-.314
Test Statistic		.314
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.39 menunjukkan hasil pengujian Kolmogorov-Smirnov Test pemakaian RAM pada *file* 4 KB (8 baris) dengan nilai signifikansi 0,000.

Pengambilan keputusan:
H0: nilai signifikansi > 0,05 maka nilai residual berdistribusi normal
H1: nilai signifikansi < 0,05 maka nilai residual tidak berdistribusi normal

Dari hasil pengujian Kolmogorov Smirnov diatas, nilai signifikansi yang dihasilkan oleh nilai residual waktu enkripsi, dekripsi, *share*, *reconstruct*, CPU, dan RAM < 0,05 yang berarti keputusan H0 ditolak karena nilai residual tidak berdistribusi normal. Sedangkan nilai signifikansi yang dihasilkan oleh nilai

residual proses enkripsi *share*, proses *reconstruct* dekripsi, dan semua proses > 0.05 yang berarti keputusan H0 diterima karena nilai residual berdistribusi normal. Apabila nilai residual berdistribusi normal, maka langkah selanjutnya dalah melakukan pengujian Anova untuk membandingkan nilai rata-rata dari semua kelompok data. Apabila nilai residual berdistribusi tidak normal, maka langkah selanjutnya adalah melakukan pengujian Kruskal-Wallis untuk mencari perbedaan waktu antar data.

6.8.1.3 Pengujian File 4KB (13 baris)

Tabel 6.40 Pengujian Kolmogorov-Smirnov pada enkripsi *file* (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		enkripsi
N		15
Normal Parameters[a,b]	Mean	.0814
	Std. Deviation	.01055
Most Extreme Differences	Absolute	.257
	Positive	.257
	Negative	-.240
Test Statistic		.257
Asymp. Sig. (2-tailed)		.009[c]

Pada Tabel 6.40 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu enkripsi *file* pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,009.

Tabel 6.41 Pengujian Kolmogorov-Smirnov pada dekripsi *file* (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		dekripsi
N		15
Normal Parameters[a,b]	Mean	.0625
	Std. Deviation	.01579
Most Extreme Differences	Absolute	.236
	Positive	.236
	Negative	-.171
Test Statistic		.236
Asymp. Sig. (2-tailed)		.024[c]

Pada Tabel 6.41 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu dekripsi *file* pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,024.

Tabel 6.42 Pengujian Kolmogorov-Smirnov *share file* (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		share
N		15
Normal Parameters[a,b]	Mean	.3813
	Std. Deviation	.21036
Most Extreme Differences	Absolute	.140
	Positive	.140
	Negative	-.138
Test Statistic		.140
Asymp. Sig. (2-tailed)		.200[c,d]

Pada Tabel 6.44 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu *share file* pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,200.

Tabel 6.43 Pengujian Kolmogorov-Smirnov *reconstruct* (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		reconstruct
N		15
Normal Parameters[a,b]	Mean	.0074
	Std. Deviation	.00819
Most Extreme Differences	Absolute	.350
	Positive	.350
	Negative	-.290
Test Statistic		.350
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.43 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu *reconstruct file* pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,000.

Tabel 6.44 Pengujian Kolmogorov-Smirnov proses enkripsi sampai *share* (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		enkripshare
N		15
Normal Parameters[a,b]	Mean	405.3835
	Std. Deviation	205.75651
Most Extreme Differences	Absolute	.219
	Positive	.219
	Negative	-.087
Test Statistic		.219
Asymp. Sig. (2-tailed)		.051[c]

Pada Tabel 6.44 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu proses enkripsi sampai *share file* pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,051.

Tabel 6.45 Pengujian Kolmogorov-Smirnov proses *reconstruct* sampai dekripsi (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		recondekrip
N		15
Normal Parameters[a,b]	Mean	2.4603
	Std. Deviation	1.16742
Most Extreme Differences	Absolute	.254
	Positive	.254
	Negative	-.169
Test Statistic		.254
Asymp. Sig. (2-tailed)		.010[c]

Pada Tabel 6.45 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu enkripsi *file* pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,010.

Tabel 6.46 Pengujian Kolmogorov-Smirnov proses semua proses (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		allproses
N		15
Normal Parameters[a,b]	Mean	411.3366
	Std. Deviation	205.07439
Most Extreme Differences	Absolute	.218
	Positive	.218
	Negative	-.092
Test Statistic		.218
Asymp. Sig. (2-tailed)		.053[c]

Pada Tabel 6.46 menunjukkan hasil pengujian Kolmogorov-Smirnov Test waktu semua proses pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,053.

Tabel 6.47 Pengujian Kolmogorov-Smirnov CPU (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		cpu
N		15
Normal Parameters[a,b]	Mean	30.0667
	Std. Deviation	3.45453
Most Extreme Differences	Absolute	.186
	Positive	.099
	Negative	-.186
Test Statistic		.186
Asymp. Sig. (2-tailed)		.174[c]

Pada Tabel 6.47 menunjukkan hasil pengujian Kolmogorov-Smirnov Test pemakaian CPU pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,174.

Tabel 6.48 Pengujian Kolmogorov-Smirnov RAM (13 baris)

One-Sample Kolmogorov-Smirnov Test		
		ram
N		25
Normal Parameters[a,b]	Mean	424.8040
	Std. Deviation	49.85571
Most Extreme Differences	Absolute	.341
	Positive	.242
	Negative	-.341
Test Statistic		.341
Asymp. Sig. (2-tailed)		.000[c]

Pada Tabel 6.48 menunjukkan hasil pengujian Kolmogorov-Smirnov Test pemakaian RAM pada *file* 4 KB (13 baris) dengan nilai signifikansi 0,000.

Pengambilan keputusan:
H0: nilai signifikansi > 0,05 maka nilai residual berdistribusi normal
H1: nilai signifikansi < 0,05 maka nilai residual tidak berdistribusi normal

Dari hasil pengujian Kolmogorov Smirnov diatas, nilai signifikansi yang dihasilkan oleh nilai residual waktu enkripsi, dekripsi, *share*, *reconstruct*, dan RAM < 0,05 yang berarti keputusan H0 ditolak karena nilai residual tidak berdistribusi normal. Sedangkan nilai signifikansi yang dihasilkan oleh nilai residual proses *write* file, proses enkripsi *share*, proses *reconstruct* dekripsi, semua proses, dan CPU > 0,05 yang berarti keputusan H0 diterima karena nilai residual berdistribusi normal. Apabila nilai residual berdistribusi normal, maka langkah selanjutnya dalah melakukan pengujian Anova untuk membandingkan nilai
rata-rata dari semua kelompok data. Apabila nilai residual berdistribusi tidak normal, maka langkah selanjutnya adalah melakukan pengujian Kruskal-Wallis untuk mencari perbedaan waktu antar data.

6.8.2 Pengujian Anova

Pengujian Anova dilakukan dengan tujuan untuk membandingkan rata-rata variabel yang terkait di semua kelompok data. Setelah membandingkan rata-rata terdapat pengujian tes homogenitas, apabila hasil signifikansi < 0,05 maka akan dilanjutkan pengujian Kruskal-Wallis, sebaliknya apabila signifikansi > 0,05 dilanjutkan pengujian Post Hoc yang menggunakan uji Bonferroni. Pengujian dilakukan dengan menganalisis nilai residual dari keseluruhan data *share* 5 *reconstruct* 2, *share* 5 *reconstruct* 3 dan *share* 5 *reconstruct* 4 dengan jumlah 15 data. Proses yang akan diuji adalah waktu enkripsi, dekripsi, *share*, *reconstruct*, proses enkripsi sampai *share*, proses *reconstruct* sampai dekripsi, semua proses, serta banyaknya pemakaian CPU dan RAM.

6.8.2.1 Pengujian File 4KB (3 baris)

Merujuk pada sub bab 6.8.1.1 telah dihasilkan bahwa nilai residual yang menerima H0 akan dilanjutkan dengan pengujian Anova. Proses yang diuji pada *file* 4 KB dengan isi 3 baris adalah waktu dari *share*, *reconstruct*, semua proses, dan pemakaian CPU. Berikut merupakan hasil dari pengujian Anova pada *file* 4 KB (3 baris) :

- **Pengujian pada *share file***

Tabel 6.49 Pengujian Anova (deskripsi) pada *share file* (3 baris)

Descriptives

share

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	.1342	.05831	.02608	.0618	.2066	.06	.22
5/3	5	.1468	.02636	.01179	.1141	.1795	.11	.17
5/4	5	.1250	.04554	.02036	.0685	.1815	.08	.19
Total	15	.1353	.04299	.01110	.1115	.1591	.06	.22

Pada Tabel 6.49 merupakan tabel deskripsi dari pengujian *share file* yang mendapatkan hasil bahwa rata-rata kombinasi *share* dan *reconstruct* menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 0.1342, *share* 5 *reconstruct* 3 memiliki rata-rata 0.1468, *share* 5 *reconstruct* 4 memiliki rata-rata 0.1353.

Tabel 6.50 Pengujian Anova (*Test Of Homogeneity of Variances*) pada *share file* (3 baris)

Test of Homogeneity of Variances

share

Levene Statistic	df1	df2	Sig.
1.288	2	12	.311

Pada Tabel 6.50 merupakan pengujian homogenitas pada proses *reconstruct* sampai dekripsi. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang diuji merupakan data yang sama (homogen).

Tabel 6.51 Pengujian Anova pada *share file* (3 baris)

ANOVA

share

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.001	2	.001	.291	.752
Within Groups	.025	12	.002		
Total	.026	14			

Pada Tabel 6.51 untuk mengetahui apakah ketiga sampel pada *share file* memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.52 Pengujian Anova (hasil Post Hoc) pada *share file* (3 baris)

Multiple Comparisons

Dependent Variable: share

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	-.01260	.02868	1.000	-.0923	.0671
	5/4	.00920	.02868	1.000	-.0705	.0889
5/3	5/2	.01260	.02868	1.000	-.0671	.0923
	5/4	.02180	.02868	1.000	-.0579	.1015
5/4	5/2	-.00920	.02868	1.000	-.0889	.0705
	5/3	-.02180	.02868	1.000	-.1015	.0579

Pada Tabel 6.52 merupakan pengujian Post Hoc untuk melihat apakah pada *share file* memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian *share file* dari membandingkan angka kombinasi 5/2 dengan 5/3, 5/2 dengan 5/4, 5/3 dengan 5/2, 5/3 dengan5/4, 5/4 dengan 5/2, dan 5/4 dengan 5/3. Hasil bisa dilihat pada Tabel 6.52.

- **Pengujian pada proses *reconstruct* sampai dekripsi**

Tabel 6.53 Pengujian Anova (deskripsi) pada proses reconstruct sampai dekripsi (3 baris)

Descriptives

recondekrip

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	2.8094	.72302	.32334	1.9117	3.7071	2.11	3.81
5/3	5	2.4602	.67800	.30321	1.6184	3.3020	1.56	3.45
5/4	5	1.0842	.18740	.08381	.8515	1.3169	.86	1.33
Total	15	2.1179	.94073	.24290	1.5970	2.6389	.86	3.81

Pada Tabel 6.53 merupakan tabel deskripsi dari pengujian pada proses *reconstruct* sampai dekripsi yang mendapatkan hasil bahwa rata-rata kombinasi

share dan *reconstruct* menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 2.8094, *share* 5 *reconstruct* 3 memiliki rata-rata 2,4602, dan *share* 5 *reconstruct* 4 memiliki rata-rata 1,0842.

Tabel 6.54 Pengujian Anova (*Test Of Homogeneity of Variances*) pada proses *reconstruct* sampai dekripsi (3 baris)

Test of Homogeneity of Variances

recondekrip

Levene Statistic	df1	df2	Sig.
2.536	2	12	.121

Pada Tabel 6.54 merupakan pengujian homogenitas pada proses *reconstruct* sampai dekripsi. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang diuji merupakan data yang sama (homogen).

Tabel 6.55 Pengujian Anova pada proses *reconstruct* sampai dekripsi (3 baris)

ANOVA

recondekrip

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	8.319	2	4.160	12.264	.001
Within Groups	4.070	12	.339		
Total	12.390	14			

Pada Tabel 6.55 untuk mengetahui apakah ketiga sampel proses *reconstruct* sampai dekripsi memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi < 0,05 yang berarti ketiga sampel memiliki rata-rata waktu yang berbeda.

Tabel 6.56 Pengujian Anova (hasil Post Hoc) pada proses *reconstruct* sampai dekripsi (3 baris)

Multiple Comparisons

Dependent Variable: recondekrip

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	.34920	.36834	1.000	-.6746	1.3730
	5/4	1.72520*	.36834	.002	.7014	2.7490
5/3	5/2	-.34920	.36834	1.000	-1.3730	.6746
	5/4	1.37600*	.36834	.009	.3522	2.3998
5/4	5/2	-1.72520*	.36834	.002	-2.7490	-.7014
	5/3	-1.37600*	.36834	.009	-2.3998	-.3522

Pada Tabel 6.56 merupakan pengujian Post Hoc untuk melihat apakah pada proses *reconstruct* sampai dekripsi memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa ada perbedaan waktu pada pengujian *share file* dari membandingkan angka kombinasi 5/2 dengan 5/3, 5/2 dengan 5/4, 5/3 dengan 5/2, 5/3 dengan5/4, 5/4 dengan 5/2, dan 5/4 dengan 5/3.

- **Pengujian pada semua proses**

Tabel 6.57 Pengujian Anova (deskripsi) pada semua proses (3 baris)

Descriptives

allproses

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	9.7374	1.33536	.59719	8.0793	11.3955	8.23	11.28
5/3	5	8.8816	1.44520	.64631	7.0872	10.6760	7.44	10.83
5/4	5	7.7970	1.32387	.59205	6.1532	9.4408	6.92	10.11
Total	15	8.8053	1.51078	.39008	7.9687	9.6420	6.92	11.28

Pada Tabel 6.57 merupakan tabel *descriptives* dari pengujian pada semua proses yang mendapatkan hasil bahwa rata-rata kombinasi *share* dan *reconstruct* menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 9,7374, *share* 5 *reconstruct* 3 memiliki rata-rata 8,8816, dan *share* 5 *reconstruct* 4 memiliki rata-rata 7,7970.

Tabel 6.58 Pengujian Anova (*Test Of Homogeneity of Variances*) pada semua proses (3 baris)

Test of Homogeneity of Variances

allproses

Levene Statistic	df1	df2	Sig.
.247	2	12	.785

Pada Tabel 6.58 merupakan pengujian homogenitas pada semua proses. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.59 Pengujian Anova pada semua proses (3 baris)

ANOVA

allproses

	Sum of Squares	Df	Mean Square	F	Sig.
Between Groups	9.457	2	4.728	2.522	.122
Within Groups	22.498	12	1.875		
Total	31.954	14			

Pada Tabel 6.59 merupakan pengujian Anova untuk mengetahui apakah ketiga sampel pada semua proses memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang berbeda.

Tabel 6.60 Pengujian Anova (hasil Post Hoc) pada semua proses (3 baris)

Multiple Comparisons

Dependent Variable: allproses

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	.85580	.86598	1.000	-1.5512	3.2628
	5/4	1.94040	.86598	.134	-.4666	4.3474
5/3	5/2	-.85580	.86598	1.000	-3.2628	1.5512
	5/4	1.08460	.86598	.703	-1.3224	3.4916
5/4	5/2	-1.94040	.86598	.134	-4.3474	.4666
	5/3	-1.08460	.86598	.703	-3.4916	1.3224

Pada Tabel 6.60 merupakan pengujian Post Hoc untuk melihat apakah pada semua proses memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian semua proses pada kombinasi 5/2 dengan 5/3, 5/2 dengan 5/4, 5/3 dengan 5/2, 5/3 dengan5/4, 5/4 dengan 5/2, dan 5/4 dengan 5/3.

- **Pengujian pada CPU**

Tabel 6.61 Pengujian Anova (dekripsi) CPU (3 baris)

Descriptives

cpu

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	6.0200	1.58965	.71091	4.0462	7.9938	4.60	8.70
5/3	5	6.4800	1.11221	.49739	5.0990	7.8610	5.20	7.70
5/4	5	5.3200	.81363	.36387	4.3097	6.3303	4.30	6.10
Total	15	5.9400	1.22812	.31710	5.2599	6.6201	4.30	8.70

Pada Tabel 6.61 merupakan tabel *descriptives* dari pengujian CPU yang mendapatkan hasil bahwa rata-rata kombinasi *share* dan *reconstruct* menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 6,0200, *share* 5

reconstruct 3 memiliki rata-rata 6,4800, dan *share* 5 *reconstruct* 4 memiliki rata-rata 5,3200.

Tabel 6.62 Pengujian Anova (*Test Of Homogeneity of Variances*) pada CPU (3 baris)

Test of Homogeneity of Variances

cpu

Levene Statistic	df1	df2	Sig.
.432	2	12	.659

Pada Tabel 6.62 merupakan pengujian homogenitas pada CPU. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.63 Pengujian Anova CPU (3 baris)

ANOVA

cpu

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	3.412	2	1.706	1.156	.347
Within Groups	17.704	12	1.475		
Total	21.116	14			

Pada Tabel 6.63 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada pengujian CPU memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.64 Pengujian Anova (Post Hoc) CPU (3 baris)

Multiple Comparisons

Dependent Variable: cpu

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	-.46000	.76820	1.000	-2.5952	1.6752
	5/4	.70000	.76820	1.000	-1.4352	2.8352
5/3	5/2	.46000	.76820	1.000	-1.6752	2.5952
	5/4	1.16000	.76820	.471	-.9752	3.2952
5/4	5/2	-.70000	.76820	1.000	-2.8352	1.4352
	5/3	-1.16000	.76820	.471	-3.2952	.9752

Pada Tabel 6.64 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian CPU memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata

waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian CPU.

6.8.2.2 Pengujian File 4KB (8 baris)

Merujuk pada sub bab 6.8.1.2 telah dihasilkan bahwa nilai residual yang menerima H0 akan dilanjutkan dengan pengujian Anova. Proses yang diuji pada *file* 4 KB dengan isi 8 baris adalah waktu dari proses enkripsi sampai *share*, proses *reconstruct* dekripsi, dan semua proses. Berikut merupakan hasil dari pengujian Anova pada *file* 4 KB (8 baris) :

- **Pengujian pada proses enkripsi sampai *share***

Tabel 6.65 Pengujian Anova (deskripsi) proses enkripsi sampai *share* (8 baris)

Descriptives

enkripshare

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	51.4104	23.76896	10.62980	21.8973	80.9235	18.50	81.66
5/3	5	77.2662	74.82567	33.46306	-15.6421	170.1745	15.81	205.14
5/4	5	68.3412	43.13798	19.29189	14.7783	121.9041	18.13	116.39
Total	15	65.6726	49.15256	12.69114	38.4528	92.8924	15.81	205.14

Pada Tabel 6.65 merupakan tabel *descriptives* dari pengujian pada proses enkripsi sampai *share* yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 51,4104, *share* 5 *reconstruct* 3 memiliki rata-rata 77,2662, dan *share* 5 *reconstruct* 4 memiliki rata-rata 68,3412.

Tabel 6.66 Pengujian Anova (*Test Of Homogeneity of Variances*) pada proses enkripsi sampai *share* (8 baris)

Test of Homogeneity of Variances

enkripshare

Levene Statistic	df1	df2	Sig.
1.544	2	12	.253

Pada tabel 6.66 merupakan pengujian homogenitas pada proses enkripsi sampai *share*. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.67 Pengujian Anova proses enkripsi sampai *share* (8 baris)

ANOVA					
enkripshare					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1724.717	2	862.358	.322	.731
Within Groups	32098.920	12	2674.910		
Total	33823.636	14			

Pada Tabel 6.67 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada proses enkripsi sampai *share* memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.68 Pengujian Anova(Post Hoc) proses enkripsi sampai *share* (8 baris)

Multiple Comparisons						
Dependent Variable: recondekrip						
Bonferroni						
					95% Confidence Interval	
(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
5/2	5/3	-25.85580	32.71030	1.000	-116.7732	65.0616
	5/4	-16.93080	32.71030	1.000	-107.8482	73.9866
5/3	5/2	25.85580	32.71030	1.000	-65.0616	116.7732
	5/4	8.92500	32.71030	1.000	-81.9924	99.8424
5/4	5/2	16.93080	32.71030	1.000	-73.9866	107.8482
	5/3	-8.92500	32.71030	1.000	-99.8424	81.9924

Pada Tabel 6.68 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian proses enkripsi sampai *share* memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian proses enkripsi sampai *share*.

- **Pengujian proses *reconstruct* sampai dekripsi**

Pada Tabel 6.69 merupakan tabel *descriptives* dari pengujian pada proses *reconstruct* sampai dekripsi yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 2,7778 , *share* 5 *reconstruct* 3 memiliki rata-rata 2,7066, dan *share* 5 *reconstruct* 4 memiliki rata-rata 2,1442.

Tabel 6.69 Pengujian Anova (deskripsi) proses *reconstruct* sampai dekripsi (8 baris)

Descriptives

recondekrip

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	2.7778	.88327	.39501	1.6811	3.8745	1.77	3.69
5/3	5	2.7066	.91305	.40833	1.5729	3.8403	1.39	3.89
5/4	5	2.1442	.25176	.11259	1.8316	2.4568	1.81	2.52
Total	15	2.5429	.75183	.19412	2.1265	2.9592	1.39	3.89

Tabel 6.70 Pengujian Anova (*Test Of Homogeneity of Variances*) pada proses *reconstruct* sampai dekripsi (8 baris)

Test of Homogeneity of Variances

recondekrip

Levene Statistic	df1	df2	Sig.
2.857	2	12	.097

Pada Tabel 6.70 merupakan pengujian homogenitas pada proses proses *reconstruct* sampai dekripsi. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.71 Pengujian Anova proses *reconstruct* sampai dekripsi (8 baris)

ANOVA

recondekrip

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.205	2	.602	1.077	.371
Within Groups	6.709	12	.559		
Total	7.914	14			

Pada Tabel 6.71 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada proses *reconstruct* sampai dekripsi memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.72 Pengujian Anova(Post Hoc) pada proses *reconstruct* sampai dekripsi (8 baris)

Multiple Comparisons

Dependent Variable: recondekrip

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	.07120	.47289	1.000	-1.2432	1.3856
	5/4	.63360	.47289	.615	-.6808	1.9480
5/3	5/2	-.07120	.47289	1.000	-1.3856	1.2432
	5/4	.56240	.47289	.772	-.7520	1.8768
5/4	5/2	-.63360	.47289	.615	-1.9480	.6808
	5/3	-.56240	.47289	.772	-1.8768	.7520

Pada Tabel 6.72 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian proses proses *reconstruct* sampai dekripsi memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian proses enkripsi sampai *share* pada kombinasi 5/3 dengan 5/4 dan 5/4 dengan 5/3.

- **Pengujian pada semua proses**

Tabel 6.73 Pengujian Anova (deskripsi) pada semua proses (8 baris)

Descriptives

allproses

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	57.4602	23.31259	10.42571	28.5138	86.4066	25.72	86.89
5/3	5	83.0730	74.99844	33.54032	-10.0499	176.1959	21.78	211.28
5/4	5	74.0012	43.25781	19.34548	20.2895	127.7129	23.53	122.52
Total	15	71.5115	49.16768	12.69504	44.2833	98.7396	21.78	211.28

Pada Tabel 6.73 merupakan tabel *descriptives* dari pengujian semua proses yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 57,4602 , *share* 5 *reconstruct* 3 memiliki rata-rata 83,0730, dan *share* 5 *reconstruct* 4 memiliki rata-rata 74,5115.

Tabel 6.74 Pengujian Anova (*Test Of Homogeneity of Variances*) pada semua proses (8 baris)

Test of Homogeneity of Variances			
allproses			
Levene Statistic	df1	df2	Sig.
1.558	2	12	.250

Pada Tabel 6.74 merupakan pengujian homogenitas pada semua proses. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.75 Pengujian Anova pada semua proses (8 baris)

ANOVA					
allproses					
	Sum of Squares	Df	Mean Square	F	Sig.
Between Groups	1686.530	2	843.265	.315	.736
Within Groups	32157.923	12	2679.827		
Total	33844.452	14			

Pada Tabel 6.75 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada proses *reconstruct* sampai dekripsi memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.76 Pengujian Anova(Post Hoc) pada semua proses (8 baris)

Multiple Comparisons						
Dependent Variable: allproses						
Bonferroni						
					95% Confidence Interval	
(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
5/2	5/3	-25.61280	32.74035	1.000	-116.6137	65.3881
	5/4	-16.54100	32.74035	1.000	-107.5419	74.4599
5/3	5/2	25.61280	32.74035	1.000	-65.3881	116.6137
	5/4	9.07180	32.74035	1.000	-81.9291	100.0727
5/4	5/2	16.54100	32.74035	1.000	-74.4599	107.5419
	5/3	-9.07180	32.74035	1.000	-100.0727	81.9291

Pada Tabel 6.76 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian semua proses memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian semua proses.

6.8.2.3 Pengujian File 4KB (13 baris)

Merujuk pada sub bab 6.8.1.3 telah dihasilkan bahwa nilai residual yang menerima H0 akan dilanjutkan dengan pengujian Anova. Proses yang diuji pada *file* 4 KB dengan isi 8 baris adalah waktu dari proses *share file*, enkripsi sampai *share*, semua proses dan CPU. Berikut merupakan hasil dari pengujian Anova pada *file* 4 KB (13 baris) :

- **Pengujian pada proses *share file***

Tabel 6.77 Pengujian Anova (deskripsi) *write file* (13 baris)

Descriptives

share

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	.2596	.12341	.05519	.1064	.4128	.05	.36
5/3	5	.3686	.22789	.10192	.0856	.6516	.05	.56
5/4	5	.5156	.21553	.09639	.2480	.7832	.27	.78
Total	15	.3813	.21036	.05431	.2648	.4978	.05	.78

Pada Tabel 6.77 merupakan tabel *descriptives* dari pengujian pada proses *share file* yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 0,2596, *share* 5 *reconstruct* 3 memiliki rata-rata 0,3686, dan *share* 5 *reconstruct* 4 memiliki rata-rata 0,5156.

Tabel 6.78 Pengujian Anova (*Test Of Homogeneity of Variances*) *share file* (13 baris)

Test of Homogeneity of Variances

share

Levene Statistic	df1	df2	Sig.
2.160	2	12	.158

Pada Tabel 6.78 merupakan pengujian homogenitas pada *write file*. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.79 Pengujian Anova *share file* (13 baris)

ANOVA

share

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.165	2	.083	2.179	.156
Within Groups	.454	12	.038		
Total	.620	14			

Pada Tabel 6.79 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada proses *share file* memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.80 Pengujian Anova (Post Hoc) *share file* (13 baris)

Multiple Comparisons

Dependent Variable: share

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	-.10900	.12308	1.000	-.4511	.2331
	5/4	-.25600	.12308	.179	-.5981	.0861
5/3	5/2	.10900	.12308	1.000	-.2331	.4511
	5/4	-.14700	.12308	.766	-.4891	.1951
5/4	5/2	.25600	.12308	.179	-.0861	.5981
	5/3	.14700	.12308	.766	-.1951	.4891

Pada Tabel 6.80 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian *share file* memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian *share file*.

- **Pengujian pada proses enkripsi sampai *share***

Tabel 6.81 Pengujian Anova (deskripsi) proses enkripsi sampai *share* (13 baris)

Descriptives

enkripshare

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	324.2452	101.05215	45.19190	198.7724	449.7180	191.50	412.66
5/3	5	425.1558	209.64579	93.75645	164.8462	685.4654	217.72	675.76
5/4	5	466.7496	283.87849	126.95432	114.2679	819.2313	122.52	905.09
Total	15	405.3835	205.75651	53.12610	291.4394	519.3277	122.52	905.09

Pada Tabel 6.81 merupakan tabel *descriptives* dari pengujian pada proses enkripsi sampai *share* yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 324,2452, *share* 5 *reconstruct* 3 memiliki rata-rata 425,1558, dan *share* 5 *reconstruct* 4 memiliki rata-rata 466,7496.

Tabel 6.82 Pengujian Anova (*Test Of Homogeneity of Variances*) pada proses enkripsi sampai *share* (13 baris)

Test of Homogeneity of Variances			
enkripshare			
Levene Statistic	df1	df2	Sig.
1.195	2	12	.336

Pada Tabel 6.82 merupakan pengujian homogenitas pada proses enkripsi sampai *share*. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.83 Pengujian Anova pada proses enkripsi sampai *share* (13 baris)

ANOVA					
enkripshare					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	53700.829	2	26850.415	.598	.566
Within Groups	538999.553	12	44916.629		
Total	592700.382	14			

Pada Tabel 6.83 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada proses enkripsi sampai *share* memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.84 Pengujian Anova (Post Hoc) pada proses enkripsi sampai *share* (13 baris)

Multiple Comparisons						
Dependent Variable: enkripshare						
Bonferroni						
		Mean			95% Confidence Interval	
(I) Shamir	(J) Shamir	Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
5/2	5/3	-100.91060	134.03974	1.000	-473.4705	271.6493
	5/4	-142.50440	134.03974	.926	-515.0643	230.0555
5/3	5/2	100.91060	134.03974	1.000	-271.6493	473.4705
	5/4	-41.59380	134.03974	1.000	-414.1537	330.9661
5/4	5/2	142.50440	134.03974	.926	-230.0555	515.0643
	5/3	41.59380	134.03974	1.000	-330.9661	414.1537

Pada Tabel 6.84 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian enkripsi sampai *share* memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian proses enkripsi sampai *share*.

- **Pengujian pada semua proses**

Tabel 6.85 Pengujian Anova (deskripsi) pada semua proses (13 baris)

Descriptives

allproses

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	331.7258	99.93436	44.69200	207.6409	455.8107	201.41	419.19
5/3	5	430.7994	209.38085	93.63796	170.8187	690.7801	223.88	681.29
5/4	5	471.4846	283.63529	126.84556	119.3049	823.6643	127.71	909.58
Total	15	411.3366	205.07439	52.94998	297.7702	524.9030	127.71	909.58

Pada Tabel 6.85 merupakan tabel *descriptives* dari pengujian pada semua proses yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 331,7258, *share* 5 *reconstruct* 3 memiliki rata-rata 430,7994, dan *share* 5 *reconstruct* 4 memiliki rata-rata 471,4846.

Tabel 6.86 Pengujian Anova (*Test Of Homogeneity of Variances*) pada semua proses (13 baris)

Test of Homogeneity of Variances

allproses

Levene Statistic	df1	df2	Sig.
1.210	2	12	.332

Pada Tabel 6.86 merupakan pengujian homogenitas pada semua proses. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.87 Pengujian Anova pada semua proses (13 baris)

ANOVA

allproses

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	51672.310	2	25836.155	.577	.576
Within Groups	537104.769	12	44758.731		
Total	588777.078	14			

Pada Tabel 6.87 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada semua proses memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.88 Pengujian Anova (Post Hoc) pada semua proses (13 baris)

Multiple Comparisons

Dependent Variable: allproses

Bonferroni

(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
5/2	5/3	-99.07360	133.80393	1.000	-470.9780	272.8308
	5/4	-139.75880	133.80393	.951	-511.6632	232.1456
5/3	5/2	99.07360	133.80393	1.000	-272.8308	470.9780
	5/4	-40.68520	133.80393	1.000	-412.5896	331.2192
5/4	5/2	139.75880	133.80393	.951	-232.1456	511.6632
	5/3	40.68520	133.80393	1.000	-331.2192	412.5896

Pada Tabel 6.88 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian semua proses memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian semua proses.

- **Pengujian pada CPU**

Tabel 6.89 Pengujian Anova (deskripsi) CPU (13 baris)

Descriptives

cpu

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
5/2	5	29.3600	2.18929	.97908	26.6416	32.0784	27.80	33.20
5/3	5	28.7600	4.73740	2.11863	22.8777	34.6423	23.40	33.50
5/4	5	32.0800	2.58979	1.15819	28.8644	35.2956	28.70	35.40
Total	15	30.0667	3.45453	.89196	28.1536	31.9797	23.40	35.40

Pada Tabel 6.89 merupakan tabel *descriptives* dari pengujian pada CPU yang mendapatkan hasil rata-rata kombinasi *share* dan *reconstruct* yang menunjukkan bahwa *share* 5 *reconstruct* 2 memiliki rata-rata 29,3600, *share* 5 *reconstruct* 3 memiliki rata-rata 28,7600, dan *share* 5 *reconstruct* 4 memiliki rata-rata 32,0800.

Tabel 6.90 Pengujian Anova (*Test Of Homogeneity of Variances*) pada CPU (13 baris)

Test of Homogeneity of Variances			
cpu			
Levene Statistic	df1	df2	Sig.
3.638	2	12	.058

Pada Tabel 6.90 merupakan pengujian homogenitas pada CPU. Hasil menunjukkan bahwa signifikansi > 0,05 yang berarti data yang di uji merupakan data yang sama (homogen).

Tabel 6.91 Pengujian Anova CPU (13 baris)

ANOVA					
cpu					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	31.301	2	15.651	1.383	.288
Within Groups	135.772	12	11.314		
Total	167.073	14			

Pada Tabel 6.91 merupakan pengujian Anova untuk menguji apakah ketiga sampel pada CPU memiliki rata-rata yang berbeda. Hasil menunjukkan bahwa nilai signifikansi > 0,05 yang berarti ketiga sampel memiliki rata-rata yang sama.

Tabel 6.92 Pengujian Anova (Post Hoc) CPU (13 baris)

Multiple Comparisons						
Dependent Variable: cpu						
Bonferroni						
					95% Confidence Interval	
(I) Shamir	(J) Shamir	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
5/2	5/3	.60000	2.12738	1.000	-5.3130	6.5130
	5/4	-2.72000	2.12738	.676	-8.6330	3.1930
5/3	5/2	-.60000	2.12738	1.000	-6.5130	5.3130
	5/4	-3.32000	2.12738	.434	-9.2330	2.5930
5/4	5/2	2.72000	2.12738	.676	-3.1930	8.6330
	5/3	3.32000	2.12738	.434	-2.5930	9.2330

Pada Tabel 6.92 merupakan pengujian Post Hoc untuk menguji apakah pada pengujian semua proses memiliki perbedaan rata-rata di setiap kelompok kombinasi. Apabila nilai signifikansi > 0,05 dapat diartikan tidak terjadi perbedaan rata-rata waktu. Hasil menunjukkan bahwa tidak ada perbedaan waktu pada pengujian CPU.

6.8.3 Pengujian Kruskal-Wallis

Pengujian Kruskal-Wallis dilakukan setelah hasil dari pengujian Kolmogorov Smirnov yang memiliki nilai residual tidak berdistribusi normal. Pengujian dilakukan pada data *share* 5 *reconstruct* 2, *share* 5 *reconstruct* 3 dan *share* 5 *reconstruct* 4 dengan jumlah data 5 berdasarkan waktu enkripsi, dekripsi, *share*, *reconstruct*, proses enkripsi sampai *share*, proses *reconstruct* sampai dekripsi, semua proses, serta banyaknya pemakaian CPU dan RAM. Langkah awal dalam pengujian Kruskal-Wallis yaitu menguji varibilitas nilai ujian data variansi algoritme Shamir's. Varibilitas yang dimaksud adalah bentuk dan sebaran data. Apabila sebaran data sama maka dapat mengetahui perbedaan *Median* dan *Mean*. Jika tidak memiliki sebaran data yang sama maka hanya dapat mengetahui perbedaan *Mean* saja. Dalam hal ini, hasil yang didapatkan adalah data variansi algoritme Shamir's tidak memiliki sebaran data yang sama sehingga hanya dapat mengetahui perbedaan nilai *Mean*.

6.8.3.1 Pengujian File 4KB (3 baris)

Merujuk pada sub bab 6.8.1.1 telah dihasilkan bahwa nilai residual berdistribusi normal maka dilanjutkan dengan pengujian Kruskal Wallis. Berikut merupakan peringkat rata-rata waktu enkripsi terdapat pada Tabel 6.93, waktu dekripsi pada Tabel 6.94, waktu *reconstruct* pada Tabel 6.95, waktu proses enkripsi sampai *share* Tabel 6.96, dan pemakaian RAM pada Tabel 6.97.

Tabel 6.93 Pengujian Kruskal-Wallis (*ranks*) enkripsi (3 baris)

Ranks			
	Shamir	N	Mean Rank
Enkripsi	5/2	5	6.90
	5/3	5	9.50
	5/4	5	7.60
	Total	15	

Tabel 6.94 Pengujian Kruskal-Wallis (*ranks*) dekripsi (3 baris)

Ranks			
	Shamir	N	Mean Rank
dekripsi	5/2	5	8.50
	5/3	5	10.30
	5/4	5	5.20
	Total	15	

Tabel 6.95 Pengujian Kruskal-Wallis (*ranks*) *reconstruct* (3 baris)

Ranks			
	Shamir	N	Mean Rank
Reconstruct	5/2	5	8.00
	5/3	5	8.00
	5/4	5	8.00
	Total	15	

Tabel 6.96 Pengujian Kruskal-wallis (*ranks*) proses enkrip sampai *share* (3 baris)

Ranks			
	Shamir	N	Mean Rank
enkrip*share*	5/2	5	7.60
	5/3	5	9.00
	5/4	5	7.40
	Total	15	

Tabel 6.97 Pengujian Kruskal-wallis (*ranks*) RAM (3 baris)

Ranks			
	Shamir	N	Mean Rank
ram	5/2	5	11.00
	5/3	5	4.60
	5/4	5	8.40
	Total	15	

Dalam menganalisis Kruskal-Wallis terdapat pengambilan keputusan berdasarkan nilai signifikansi yang diperoleh untuk mengukur perbedaan peringkat rata-rata yang didapatkan signifikan atau tidak. Hasil pengujian Kruskal Wallis waktu enkripsi terdapat pada Tabel 6.98, waktu dekripsi pada Tabel 6.99, waktu *reconstruct* pada Tabel 6.100, waktu proses enkripsi sampai *share* Tabel 6.101, dan pemakaian RAM pada Tabel 6.102.

Tabel 6.98 Pengujian Kruskal-Wallis (*statistic*) enkripsi (3 baris)

Test Statistics[a,b]	
	Enkripsi
Chi-Square	1.181
df	2
Asymp. Sig.	.554
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.99 Pengujian Kruskal-Wallis (*statistic*) dekripsi (3 baris)

Test Statistics[a,b]	
	Dekripsi
Chi-Square	3.739
df	2
Asymp. Sig.	.154
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.100 Pengujian Kruskal-Wallis (*statistic*) *reconstruct* (3 baris)

Test Statistics[a,b]	
	reconstruct
Chi-Square	.000
df	2
Asymp. Sig.	1.000
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.101 Pengujian Kruskal-Wallis (*statistic*) proses enkrip sampai *share* (3 baris)

Test Statistics[a,b]	
	enkripshare
Chi-Square	.380
Df	2
Asymp. Sig.	.827
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.102 Pengujian Kruskal-Wallis (*statistic*) RAM (3 baris)

Test Statistics[a,b]	
	ram
Chi-Square	5.189
df	2
Asymp. Sig.	.075
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Pengambilan keputusan:

H0: signifikansi > 0,05 maka tidak terdapat perbedaan waktu antar data kelompok

H1: signifikansi < 0,05 maka terdapat perbedaan waktu antar data kelompok

Hasil yang didapatkan dari Tabel 6.98 sampai Tabel 6.102 adalah terdapat nilai signifikansi > 0,05 maka keputusan H0 diterima yang berarti tidak terdapat perbedaan waktu yang signifikan antara data kombinasi *share* dan *reconstruct* yang dianalisis sehingga tidak memerlukan pengujian Post Hoc.

6.8.3.2 Pengujian File 4KB (8 baris)

Merujuk pada sub bab 6.8.1.2 telah dihasilkan bahwa nilai residual berdistribusi normal maka dilanjutkan dengan pengujian Kruskal Wallis. Berikut merupakan hasil pengujian pada *file* 4 KB (8 baris) dengan peringkat rata-rata waktu enkripsi terdapat pada Tabel 6.103, waktu dekripsi pada Tabel 6.104, waktu *share* pada Tabel 6.105, waktu *reconstruct* pada Tabel 6.106, pemakaian CPU pada Tabel 6.107, dan pemakaian RAM pada Tabel 6.108.

Tabel 6.103 Pengujian Kruskal-wallis (*ranks*) enkripsi (8 baris)

Ranks			
	Shamir	N	Mean Rank
enkripsi	5/2	5	8.80
	5/3	5	7.70
	5/4	5	7.50
	Total	15	

Tabel 6.104 Pengujian Kruskal-wallis (*ranks*) dekripsi (8 baris)

Ranks			
	Shamir	N	Mean Rank
dekripsi	5/2	5	10.60
	5/3	5	9.90
	5/4	5	3.50
	Total	15	

Tabel 6.105 Pengujian Kruskal-wallis (*ranks*) *share* (8 baris)

Ranks			
	Shamir	N	Mean Rank
share	5/2	5	11.60
	5/3	5	8.30
	5/4	5	4.10
	Total	15	

Tabel 6.106 Pengujian Kruskal-wallis (*ranks*) *reconstruct* (8 baris)

Ranks			
	Shamir	N	Mean Rank
reconstruct	5/2	5	6.70
	5/3	5	8.90
	5/4	5	8.40
	Total	15	

Tabel 6.107 Pengujian Kruskal-wallis (*ranks*) CPU (8 baris)

Ranks			
	Shamir	N	Mean Rank
cpu	5/2	5	10.80
	5/3	5	9.40
	5/4	5	3.80
	Total	15	

Tabel 6.108 Pengujian Kruskal-wallis (*ranks*) RAM (8 baris)

Ranks			
	Shamir	N	Mean Rank
ram	5/2	5	10.40
	5/3	5	4.00
	5/4	5	9.60
	Total	15	

Setelah itu dalam menganalisis Kruskal-Wallis terdapat pengambilan keputusan berdasarkan nilai signifikansi yang diperoleh untuk mengukur perbedaan peringkat rata-rata yang didapatkan signifikan atau tidak. Hasil pengujian Kruskal Wallis waktu enkripsi terdapat pada Tabel 6.109, waktu dekripsi pada Tabel 6.110, waktu *share* pada Tabel 6.111, waktu *reconstruct* pada Tabel 6.112, pemakaian CPU pada Tabel 6.113, dan pemakaian RAM pada Tabel 6.114.

Tabel 6.109 Pengujian Kruskal-wallis (*statistic*) enkripsi (8 baris)

Test Statistics[a,b]	
	enkripsi
Chi-Square	.501
df	2
Asymp. Sig.	.779
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.110 Pengujian Kruskal-wallis (*statistic*) dekripsi (8 baris)

Test Statistics[a,b]	
	dekripsi
Chi-Square	8.340
df	2
Asymp. Sig.	.015
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.111 Pengujian Kruskal-wallis (*statistic*) *share* (8 baris)

Test Statistics[a,b]	
	share
Chi-Square	7.116
df	2
Asymp. Sig.	.028
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.112 Pengujian Kruskal-wallis (*statistic*) *reconstruct* (8 baris)

Test Statistics[a,b]	
	reconstruct
Chi-Square	.955
df	2
Asymp. Sig.	.620
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.113 Pengujian Kruskal-wallis (*statistic*) CPU (8 baris)

Test Statistics[a,b]	
	cpu
Chi-Square	6.860
df	2
Asymp. Sig.	.032
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.114 Pengujian Kruskal-wallis (*statistic*) RAM (8 baris)

Test Statistics[a,b]	
	ram
Chi-Square	6.080
df	2
Asymp. Sig.	.048
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Pengambilan keputusan:

H0: signifikansi > 0,05 maka tidak terdapat perbedaan waktu antar data kelompok

H1: signifikansi < 0,05 maka terdapat perbedaan waktu antar data kelompok

Hasil yang didapatkan adalah terdapat nilai signifikansi > 0,05 maka keputusan H0 diterima yang berarti tidak terdapat perbedaan waktu antara data kombinasi *share* dan *reconstruct* yang dianalisis. Hasil signifikansi>0,05 bisa dilihat pada Tabel 6.109 dan Tabel 6.112. Terdapat juga nilai signifikansi < 0,05 maka H1 diterima yang berarti terdapat perbedaan waktu antara data kombinasi *share* dan *reconstruct*. Hasil signifikansi<0,05 bisa dilihat pada Tabel 6.110, Tabel 6.111, Tabel 6.113 dan Tabel 6.114. Apabila H1 diterima, maka langkah selanjutnya adalah melakukan pengujian Post Hoc *Test* untuk mengetahui dimana data kelompok yang menimbulkan perbedaan waktu tertinggi.

6.8.3.3 Pengujian File 4KB (13 baris)

Merujuk pada sub bab 6.8.1.3 telah dihasilkan bahwa nilai residual berdistribusi normal maka dilanjutkan dengan pengujian Kruskal Wallis. Berikut merupakan peringkat rata-rata waktu enkripsi terdapat pada Tabel 6.115, waktu dekripsi pada Tabel 6.116, waktu *reconstruct* pada Tabel 6.117, waktu *reconstruct* sampai dekripsi pada Tabel 6.118, dan pemakaian RAM pada Tabel 6.119.

Tabel 6.115 Pengujian Kruskal-Wallis (*ranks*) enkripsi (13 baris)

Ranks			
	Shamir	N	Mean Rank
enkripsi	5/2	5	5.10
	5/3	5	8.70
	5/4	5	10.20
	Total	15	

Tabel 6.116 Pengujian Kruskal-Wallis (*ranks*) dekripsi (13 baris)

Ranks			
	Shamir	N	Mean Rank
dekripsi	5/2	5	9.70
	5/3	5	7.30
	5/4	5	7.00
	Total	15	

Tabel 6.117 Pengujian Kruskal-Wallis (*ranks*) *reconstruct* (13 baris)

Ranks			
	Shamir	N	Mean Rank
reconstruct	5/2	5	4.50
	5/3	5	10.20
	5/4	5	9.30
	Total	15	

Tabel 6.118 Pengujian Kruskal-Wallis (*ranks*) *reconstruct* sampai dekripsi (13 baris)

Ranks			
	Shamir	N	Mean Rank
recondekrip	5/2	5	12.20
	5/3	5	8.20
	5/4	5	3.60
	Total	15	

Tabel 6.119 Pengujian Kruskal-Wallis (*ranks*) RAM (13 baris)

Ranks			
	Shamir	N	Mean Rank
ram	5/2	5	5.00
	5/3	5	11.70
	5/4	5	7.30
	Total	15	

Setelah itu dalam menganalisis Kruskal-Wallis terdapat pengambilan keputusan berdasarkan nilai signifikansi yang diperoleh untuk mengukur perbedaan peringkat rata-rata yang didapatkan signifikan atau tidak. Hasil pengujian Kruskal Wallis waktu enkripsi terdapat pada Tabel 6.120, waktu dekripsi pada Tabel 6.121, waktu *reconstruct* pada Tabel 6.122, waktu *reconstruct* sampai dekripsi pada Tabel 6.123, dan pemakaian RAM pada Tabel 6.124.

Tabel 6.120 Pengujian Kruskal-Wallis (*statistic*) enkripsi (13 baris)

Test Statistics[a,b]	
	enkripsi
Chi-Square	3.983
Df	2
Asymp. Sig.	.137
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.121 Pengujian Kruskal-Wallis (*statistic*) dekripsi (13 baris)

Test Statistics[a,b]	
	dekripsi
Chi-Square	1.129
Df	2
Asymp. Sig.	.569
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.122 Pengujian Kruskal-Wallis (*statistic*) *reconstruct* (13 baris)

Test Statistics[a,b]	
	reconstruct
Chi-Square	5.962
Df	2
Asymp. Sig.	.051
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.123 Pengujian Kruskal-Wallis (*statistic*) *reconstruct* sampai dekripsi (13 baris)

Test Statistics[a,b]	
	recondekrip
Chi-Square	9.260
Df	2
Asymp. Sig.	.010
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Tabel 6.124 Pengujian Kruskal-Wallis (*statistic*) RAM (13 baris)

Test Statistics[a,b]	
	ram
Chi-Square	5.805
Df	2
Asymp. Sig.	.055
a. Kruskal Wallis Test	
b. Grouping Variable: Shamir	

Pengambilan keputusan:
H0: signifikansi > 0,05 maka tidak terdapat perbedaan waktu antar data kelompok
H1: signifikansi < 0,05 maka terdapat perbedaan waktu antar data kelompok

Hasil yang didapatkan adalah terdapat nilai signifikansi > 0,05 maka keputusan H0 diterima yang berarti tidak terdapat perbedaan waktu antara data kombinasi *share* dan *reconstruct* yang dianalisis. Hasil signifikansi>0,05 dapat dilihat pada Tabel 6.120, Tabel 6.121, Tabel 6.122, danTabel 6.124. Terdapat juga nilai signifikansi< 0,05 maka H1 diterima yang berarti terdapat perbedaan waktu antara data kombinasi *share* dan *reconstruct*. Hasil signifikansi<0,05 dapat dilihat pada Tabel 6.123. Apabila H1 diterima, maka langkah selanjutnya adalah melakukan pengujian Post Hoc *Test* untuk mengetahui dimana data kelompok yang menimbulkan perbedaan waktu tertinggi.

6.8.4 Pengujian Post Hoc *Test*

Pengujian Post Hoc *Test* dilakukan pada data kelompok dari *share* 5 *reconstruct* 2 dengan sebutan data 1, *share* 5 *reconstruct* 3 dengan sebutan data 2, *share* 5 *reconstruct* 2 dengan sebutan data 3. Pengujian ini akan membandingan data 1 dengan data 2, data 1 dengan data 3, dan data 2 dengan data 3. Pengujian ini ditentukan dari nilai signifikansi > 0,05 yang artinya terdapat perbedaan waktu antara data kelompok. Apabila signifikansi < 0,05, maka akan dilakukan pengujian Post Hoc *test* untuk mencari perbedaan waktu tertinggi diantara data kelompok yang diujikan.

6.8.4.1 Pengujian File 4KB (8 baris)

Merujuk pada sub bab 6.8.3.2 telah dihasilkan bahwa nilai residual yang menerima H1 akan dilanjutkan dengan pengujian Post Hoc *Test*. Berikut merupakan pengujian Post Hoc *Test* :

- **Pengujian Post Hoc pada proses dekripsi**
  Berikut merupakan hasil pengujian Post Hoc *Test* untuk proses dekripsi *file* dengan ukuran 4 KB (8 baris) adalah sebagai berikut:
  1. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 3 bernilai 0,120. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.125.

Tabel 6.125 Pengujian Post Hoc Test (*statistic*) dekripsi data 1 dan 2

Test Statistics[a,b]	
	Dekripsi
Chi-Square	.120
df	1
Asymp. Sig.	.729

2. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 bernilai 5,989. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.126.

Tabel 6.126 Pengujian Post Hoc Test (*statistic*) dekripsi data 1 dan 3

Test Statistics[a,b]	
	dekripsi
Chi-Square	5.989
df	1
Asymp. Sig.	.014

3. Nilai *Chi-Square* antara *share* 5 *reconstruct 3* dengan *share* 5 *reconstruct* 4 bernilai 5,694. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.127.

Tabel 6.127 Pengujian Post Hoc Test (*statistic*) dekripsi data 2 dan 3

Test Statistics[a,b]	
	dekripsi
Chi-Square	5.694
df	1
Asymp. Sig.	.017

Pada hasil pengujian Post Hoc Test dekripsi nilai *Chi-Square* tertinggi terdapat pada data kelompok *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 yang bernilai 5,989. Hasil dari nilai signifikansi pada Tabel 6.126 < 0,05 yang berarti terdapat perbedaan waktu yang signifikan. Hal ini terjadi karena perbedaan waktu dari data kelompok lebih besar dibanding dengan data kelompok lainnya.

- **Pengujian Post Hoc pada proses *share***

Berikut merupakan hasil pengujian Post Hoc *Test* untuk waktu *share file* dengan ukuran 4 KB (8 baris) adalah sebagai berikut:

1. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 3 bernilai 1,590. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.128.

Tabel 6.128 Pengujian Post Hoc Test (*statistic*) *share* data 1 dan 2

Test Statistics[a,b]	
	share
Chi-Square	1.590
df	1
Asymp. Sig.	.207

2. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 bernilai 6,361. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.129.

Tabel 6.129 Pengujian Post Hoc *Test* (*statistic*) *share* data 1 dan 3

Test Statistics[a,b]	
	share
Chi-Square	6.361
df	1
Asymp. Sig.	.012

3. Nilai *Chi-Square* antara *share* 5 *reconstruct 3* dengan *share* 5 *reconstruct* 4 bernilai 2,485. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.130.

Tabel 6.130 Pengujian Post Hoc *Test* (*statistic*) *share* data 2 dan 3

Test Statistics[a,b]	
	share
Chi-Square	2.485
df	1
Asymp. Sig.	.115

Pada hasil pengujian Post Hoc Test *share* nilai *Chi-Square* tertinggi terdapat pada data 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 yang bernilai 6,361. Hasil dari nilai signifikansi pada Tabel 6.129 < 0,05 yang berarti terdapat perbedaan waktu yang signifikan. Hal ini terjadi karena waktu yang dihasilkan tidak sama dari data kelompok dengan data kelompok lainnya.

- **Pengujian Post Hoc pada CPU**

Berikut merupakan hasil pengujian Post Hoc *Test* untuk pemakaian CPU dengan ukuran 4 KB (8 baris) adalah sebagai berikut:

1. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 3 bernilai 0,098. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.131.

Tabel 6.131 Pengujian Post Hoc *Test* (*ranks*) CPU data 1 dan 2

Test Statistics[a,b]	
	cpu
Chi-Square	.098
df	1
Asymp. Sig.	.754

2. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 bernilai 6,818. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.132.

Tabel 6.132 Pengujian Post Hoc Test (*ranks*) CPU data 1 dan 3

Test Statistics[a,b]	
	cpu
Chi-Square	6.818
df	1
Asymp. Sig.	.009

3. Nilai *Chi-Square* antara *share* 5 *reconstruct 3* dengan *share* 5 *reconstruct* 4 bernilai 3,153. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.133.

Tabel 6.133 Pengujian Post Hoc Test (*ranks*) CPU data 2 dan 3

Test Statistics[a,b]	
	cpu
Chi-Square	3.153
df	1
Asymp. Sig.	.076

Pada hasil pengujian Post Hoc Test CPU nilai *Chi-Square* tertinggi terdapat pada data *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 yang bernilai 6,818. Hasil dari nilai signifikansi pada Tabel 6.132 < 0,05 yang berarti terdapat perbedaan pemakaian yang signifikan. Hal ini terjadi karena perbedaan pemakaian yang dihasilkan lebih besar dari data kelompok dengan data kelompok lainnya.

- **Pengujian Post Hoc pada RAM**

Berikut merupakan hasil pengujian Post Hoc *Test* untuk pemakaian RAM dengan ukuran 4 KB (8 baris) adalah sebagai berikut:

1. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 3 bernilai 2,455. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.134.

Tabel 6.134 Pengujian Post Hoc Test (*ranks*) RAM data 1 dan 2

Test Statistics[a,b]

	ram
Chi-Square	2.455
df	1
Asymp. Sig.	.117

2. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 bernilai 0,884. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.135.

Tabel 6.135 Pengujian Post Hoc Test (*ranks*) RAM data 1 dan 3

Test Statistics[a,b]

	ram
Chi-Square	.884
df	1
Asymp. Sig.	.347

3. Nilai *Chi-Square* antara *share* 5 *reconstruct 3* dengan *share* 5 *reconstruct* 4 bernilai 6,818. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.136.

Tabel 6.136 Pengujian Post Hoc Test (*ranks*) RAM data 2 dan 3

Test Statistics[a,b]

	ram
Chi-Square	6.818
df	1
Asymp. Sig.	.009

Pada hasil pengujian Post Hoc Test RAM nilai *Chi-Square* tertinggi terdapat pada *share* 5 *reconstruct 3* dengan *share* 5 *reconstruct* 4 yang bernilai 6,818. Hasil dari nilai signifikansi pada Tabel 6.136 < 0,05 yang berarti terdapat perbedaan pemakaian yang signifikan. Hal ini terjadi karena perbedaan pemakaian yang dihasilkan lebih besar dari data kelompok dengan data kelompok lainnya.

6.8.4.2 Pengujian File 4KB (13 baris)

Merujuk pada sub bab 6.8.3.3 telah dihasilkan bahwa nilai residual yang menerima H1 akan dilanjutkan dengan pengujian Post Hoc *Test*. Berikut merupakan pengujian Post Hoc *Test* :

- **Pengujian Post Hoc proses *reconstruct* sampai dekripsi**
  Berikut merupakan hasil pengujian Post Hoc *Test* untuk proses *reconstruct sampai dekripsi* dengan ukuran 4 KB (13 baris) adalah sebagai berikut:

1. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 3 bernilai 3,153. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.137.

Tabel 6.137 Pengujian Post Hoc Test (*ranks*) *share* data 1 dan 2

Test Statistics[a,b]	
	recondekrip
Chi-Square	3.153
df	1
Asymp. Sig.	.076

2. Nilai *Chi-Square* antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 bernilai 6,818. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.138.

Tabel 6.138 Pengujian Post Hoc Test (*ranks*) *share* data 1 dan 3

Test Statistics[a,b]	
	recondekrip
Chi-Square	6.818
df	1
Asymp. Sig.	.009

3. Nilai *Chi-Square* antara *share* 5 *reconstruct* 3 dengan *share* 5 *reconstruct* 4 bernilai 2,917. Nilai *Chi-Square* data kelompok ditunjukkan pada Tabel 6.139.

Tabel 6.139 Pengujian Post Hoc Test (*ranks*) *share* data 2 dan 3

Test Statistics[a,b]	
	recondekrip
Chi-Square	3.938
df	1
Asymp. Sig.	.047

Pada hasil pengujian Post Hoc Test *share* nilai *Chi-Square* tertinggi terdapat pada antara *share* 5 *reconstruct* 2 dengan *share* 5 *reconstruct* 4 yang bernilai 6,818. Hasil dari nilai signifikansi pada Tabel 6.138 < 0,05 yang berarti terdapat perbedaan waktu yang signifikan. Hal ini terjadi karena perbedaan waktu yang dihasilkan lebih besar dari data kelompok lainnya.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan beberapa hal mengenai algoritme SPECK *Block Cipher* dan Shamir's *Secret Sharing* adalah sebagai berikut :

1. Algoritme SPECK dapat diterapkan pada *file* teks. Algoritme SPECK dapat memberikan aspek kerahasiaan (*confidentiality*), hal ini dapat dibuktikan bahwa algoritme SPECK dapat dibuktikan dari pengujian enkripsi dan dekripsi yang telah dilakukan dan dengan status valid. Algoritme Shamir's *Secret Sharing* dapat memecah dan mengembalikan *file* teks seperti semula dan memberikan aspek ketersediaan data (*availability*). Hal ini dapat dibuktikan dari pengujian *share* dan *reconstruct* dengan status valid.
2. Dari hasil pengujian waktu didapatkan hasil bahwa rata-rata total waktu proses enkripsi pada ukuran *file* 4 KB dengan isi 3 baris yaitu 0,033 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 0,069 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 0,074 *s*. Rata-rata total waktu dekripsi pada ukuran *file* 4 KB dengan isi 3 baris yaitu 0.022 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 0,054 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 0,065 *s*. Rata-rata total waktu *share file* pada ukuran *file* 4 KB dengan isi 3 baris yaitu 0,175 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 0,263 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 0,647*s*. Rata-rata total waktu *reconstruct* pada ukuran *file* 4 KB dengan isi 3 baris yaitu 0,002 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 0,009 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 0,012 *s*. Rata-rata total waktu proses enkripsi sampai *share* pada ukuran *file* 4 KB dengan isi 3 baris yaitu 4,200 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 82,911 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 469,056 *s*. Rata-rata total waktu proses *reconstruct* sampai dekripsi pada ukuran *file* 4 KB dengan isi 3 baris yaitu 2,248 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 3,401 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 4,861 *s*. Rata-rata total waktu semua proses pada ukuran *file* 4 KB dengan isi 3 baris yaitu 9,433 *s*, ukuran *file* 4 KB dengan isi 8 baris yaitu 91,938 *s*, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 619,036 *s*.
3. Terdapat rata-rata banyaknya pemakaian CPU pada ukuran *file* 4 KB dengan isi 3 baris yaitu 7,058%, ukuran *file* 4 KB dengan isi 8 baris yaitu 25,885%, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 31,095%. Terdapat rata-rata banyaknya pemakaian RAM pada ukuran *file* 4 KB dengan isi 3 baris yaitu 377,277 MB, ukuran *file* 4 KB dengan isi 8 baris yaitu 414,283 MB, sedangkan ukuran *file* 4 KB dengan isi 13 baris yaitu 440,231 MB.

7.2 Saran

1. Sistem hanya dapat melakukan enkripsi, dekripsi, *share* dan *reconstruct* pada file *teks*, untuk penelitian Selanjutnya diharapkan dapat melakukan proses

enkripsi, dekripsi, *share* dan *reconstruct* pada *file* video, audio maupun gambar.

2. Untuk penelitian selanjutnya diharapkan dapat diimplementasikan dengan GUI dan diberbagai *platform.*

DAFTAR REFERENSI

Ariyus, D., 2008. *Pengantar Ilmu Kriptografi : Teori analisis & implementasi..* Yogyakarta: CV ANDI OFFSET.

Beaulieu, Ray dkk, 2015. The SIMON and SPECK lightweight block ciphers.

Canavan, J. E., 2001. *Fundamentals of Network Security.* London: Artech House.

Dicky Pratama & Herry Septriadi, 2016. Pengaruh Pemanfaatan Kelas Elektronik Terhadap Efektifitas dan Efisiensi Proses Belajar STMIK XYZ.

Dino Caesaron, S. S. A. N., 2015. Pengaruh Kecepatan Putar Spindel Dalam Pengujian Viskositas Produk UQ. Black QHS Dengan Metode Anova Pelangi Chemindo.

Drs. Ario Suryo Kusumo, 2004. In: *Visual Basic.NET versi 2002 dan 2003.* Jakarta: Pt Elex Media Komputindo.

Eko Hari Rachmawanto, dkk, 2015. Keamanan File Menggunakan Teknik Kriptografi Shift Cipher.

Endro Ariyanto, Trisya Indah Pravitasari, Setyorini, 2008. Analisa Implementasi Algoritma Stream Cipher Sosemanuk Dan Dicing Dalam Proses Enkripsi Data.

Ermawati, E. F., 2010. *Efek antipiretik ekstrak daun pare (momordica charantia l.) Pada tikus putih jantan.* s.l.:s.n.

Genge, A. V. D. d. B., 2017. Implementation of SIMON and SPECK Lightweight Block Cipher on Programmablre Logic Controllers.

Lisa Chandra, Y. D. L. W. S. M. M. A. S., 2012. Penerapan Algoritma “LaGrange Interpolating Polynomial” Pada Secret Sharing.

Mustafa Ulutas, G. U. V. V. N., 2010. Medical Image Security and EPR hiding using Shamir's Secret SHaring Scheme.

Nawangsari, T., 2013. Perbandingan Berganda Sesudah Uji Kruskal-Wallis.

Shamir, A., 1979. How to Share a Secret.

Tim EMS, 2015. *Kamus Komputer Lengkap,* Jakarta: PT Elex Media Komputindo.